Machine learning and privacy

Lecturer: Dr. Ana-Maria Cretu

Slides' derivation chain:
Ana-Maria Cretu <- [Dario, Pasquini, Carmela Troncoso] <-> [Rebekah Overdorf, Bogdan Kulynych and **Theresa Stadler]**

Objectives

- “Machine learning (ML) and privacy” is an extremely active and dynamic line of research; fresh results every month!
  - Historically, there has been a focus on predictive AI, i.e., classification models.
  - More recently, the focus has shifted to generative AI: large language models (LLM), diffusion models...

- This lecture focuses on the fundamental concepts. You will learn how to:
  - Think adversarially about ML in two widely used settings, centralized ML and federated ML.
  - Train ML models in a privacy-preserving way.

Lecture overview

- What can we learn from machine learning models? (focus on deep learning)
  - Centralized machine learning (50%)
  - Collaborative machine learning (50%)

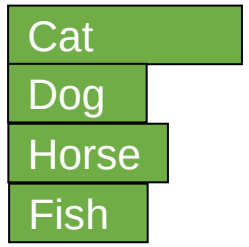
- Learning on private data
  - How can we responsibly use private data?

Part I – Centralized ML

Useful ML background

- We want to train a model to infer a label Y given a feature vector X for a distribution (X, Y) over D^*. E.g., infer a user's pet given their social network profile.
  - D^* is the universe of possible samples. E.g., all social network profiles and the owners' pets.
- A training dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ consists of n samples of (X, Y). Each label denotes one of C classes.
- **At train time:** We train a model on D.
- **Test/inference time**: We use the model to make predictions on new inputs x.

  Given x, we retrieve the model's confidences on each class, $M(x) = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C)$, $\sum_i \hat{y}_i = 1$. These are the probabilities the model assigns to the event "x belongs to the i-th class". We predict j such that $\hat{y}_j = \max_i \hat{y}_i$.



$\hat{y}_1 = 0.40$
$\hat{y}_2 = 0.18$
$\hat{y}_3 = 0.24$
$\hat{y}_4 = 0.18$

How to get a model from the data?

- By applying a training algorithm T to the training dataset D: $M = T(D, seed)$.
  - Model architecture (e.g., neural network)
  - Objective (loss) function.
- The model is specified by its architecture and trained parameters θ (also called weights): $M = M(\theta)$
- Model predictions on inputs x: $M(x) = M(x; \theta)$

Stochastic gradient descent (SGD)

Example: neural network classifier.

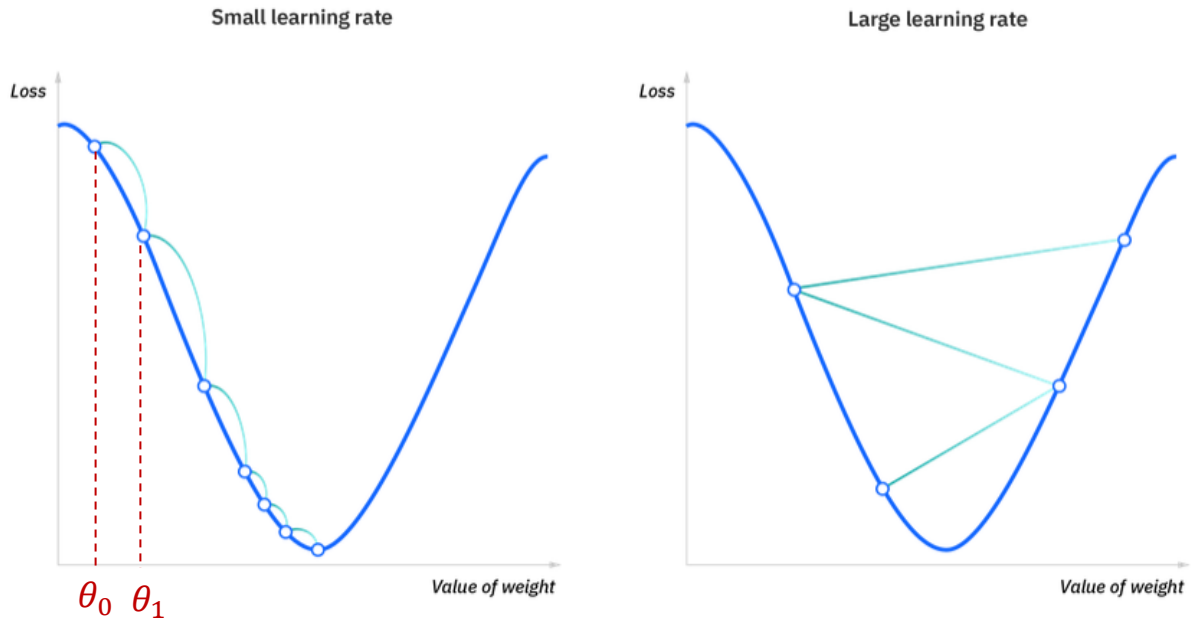
Training process (mini-batch SGD):

Randomly initialize the weights $\theta_0 \leftarrow random$.
For t in $\{0,1,\dots,n-1\}$:

1. Sample batch from the training set:
$$(x_1, y_1), \dots, (x_B, y_B) \sim D$$
2. Compute model's prediction (confidence score):
$$\widehat{y_b} = M(x_b, \theta_t), b = 1, \dots B$$
3. Compute the loss:
$$l(\theta_t) = \frac{1}{B}\sum_{b=1}^{B} L(\hat{y}_b, y_b)$$
4. Compute gradient:
$$\mathrm{g_t} = \nabla_{\theta_t} l$$
5. Update parameters:
$$\theta_{t+1} = \theta_t - \eta \mathrm{g_t}$$

D: training dataset
b: batch size
$M(\theta)$: model parameterized by θ
L: loss function
η: learning rate
n: number of training steps

Source: https://www.ibm.com/think/topics/gradient-descent



Machine learning – Life cycle

1. **Task design:** Define the objective of the model.
2. **Data collection:** Collect a dataset for training the model.
3. **Train the model:** Fit a model to the dataset, by applying a training algorithm.
4. **Deploy:** Make the model available to someone else (e.g., via APIs or weight sharing).

Machine learning – Life cycle

1. **Task design:** Define the objective of the model.
2. **Data collection:** Collect a dataset for training the model.
3. **Train the model:** Fit a model to the dataset, by applying a training algorithm.
4. **Deploy:** Make the model available to someone else (e.g., via APIs or weight sharing).

This is where the privacy problems start!

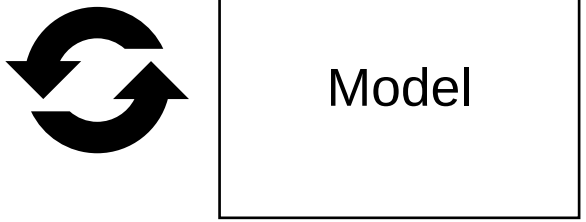
Machine learning privacy concerns

Our main concern is the confidentiality of the training set. Why?

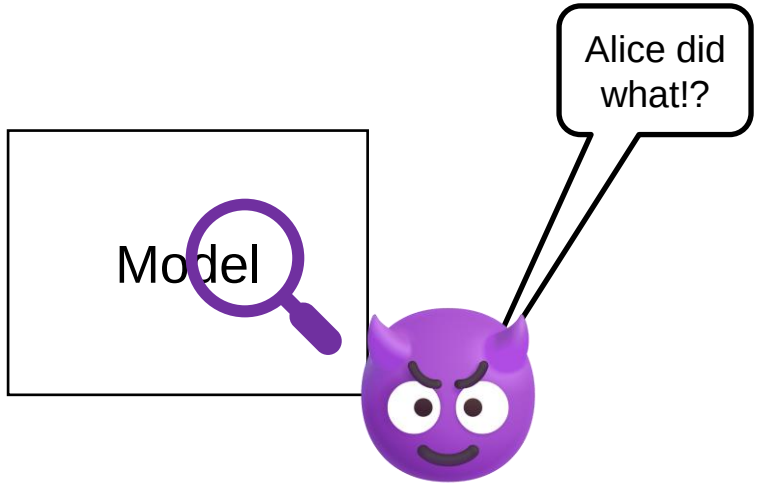
- 🔒 Training data may be sensitive!
- 💸 Training data is expensive to collect.

Training phase:

Training sample:



Deployment phase:



Objectives of privacy attacks

An adversary can target information about:

1. Individuals who contributed to the training data
   - Is a sample (person, image,…) in the training set? → **Membership inference attack**
   - If so, what attributes does this sample has? → **Attribute inference attack**
   - Can I extract training samples from the model [1]? → **Reconstruction/data extraction attack**

2. The dataset as a whole
   - What is the proportion of women that this model was trained on? → **Property inference attack**

[1]Some models like Support Vector Machines or k-Nearest Neighbors encode (a subset of) their training samples. Recovering samples from these models is straightforward given access to the parameters. In this lecture, we focus on deep neural networks trained using SGD, where the problem is non-trivial.

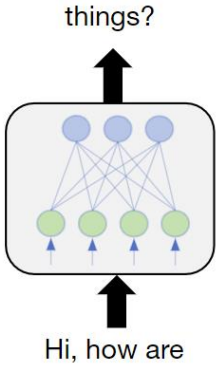
Threat model

Black-box attack

White-box attack

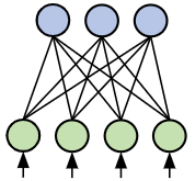
- The adversary can query the model M on any valid input x of their choice, to retrieve the model confidences[1] for each class $M(x) = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C),\ \sum_i \hat{y}_i = 1$.

- Examples: Machine Learning as a Service (MLaaS) and models accessed through APIs.



- The adversary has access to the trained model's parameters θ, i.e., complete knowledge of how the model works.

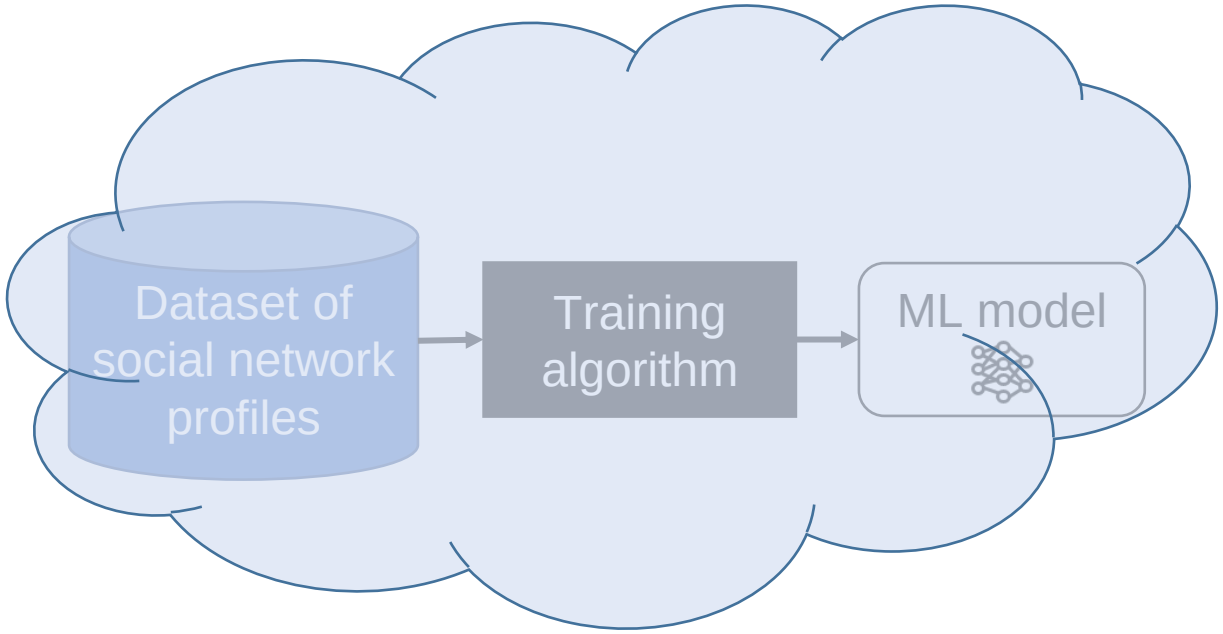
- Examples: on-device releases and open-source models.



[1]Variants of the black-box threat model include only releasing the predicted label $\arg\ \max_i\ \hat{y}_i$ and only releasing the top-k confidences.

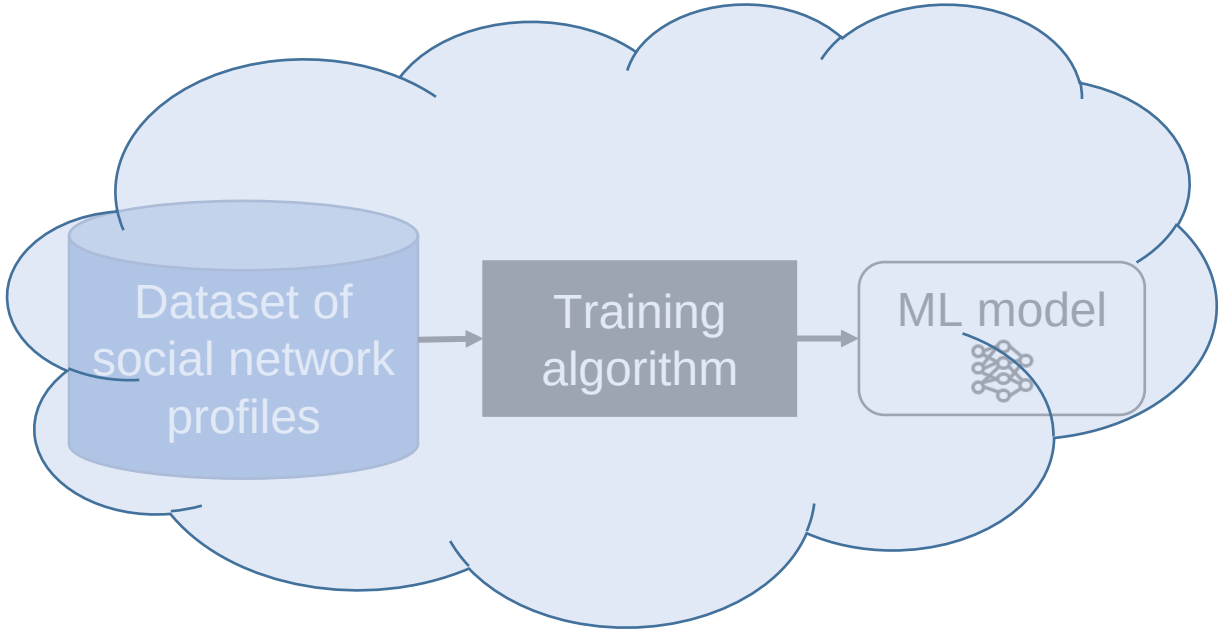
Machine Learning as a Service (MLaaS)

1. A company trains a model in the cloud.



Machine Learning as a Service (MLaaS)

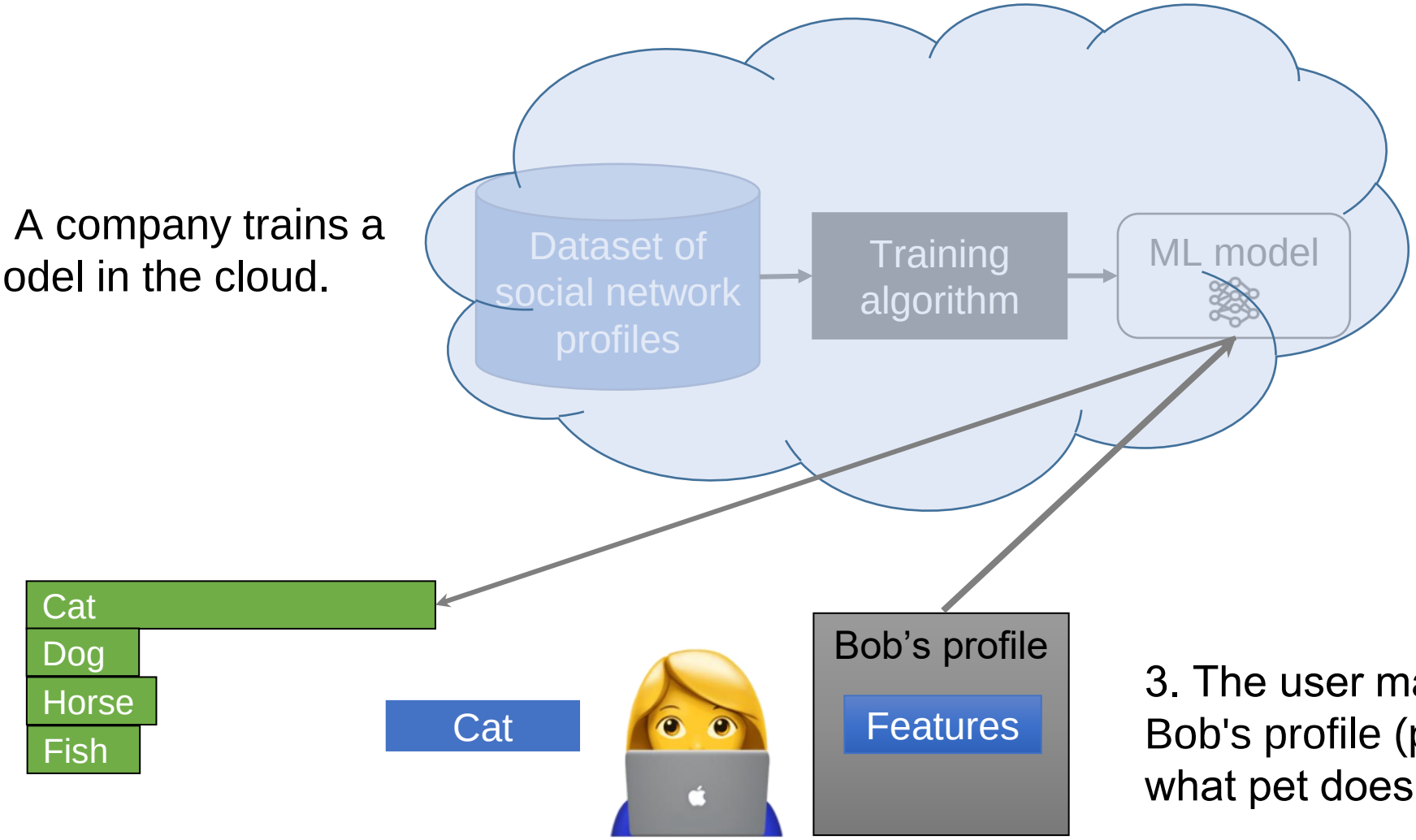
1. A company trains a model in the cloud.



2. The company makes this model available as a service for users to query

Machine Learning as a Service (MLaaS)

1. A company trains a model in the cloud.

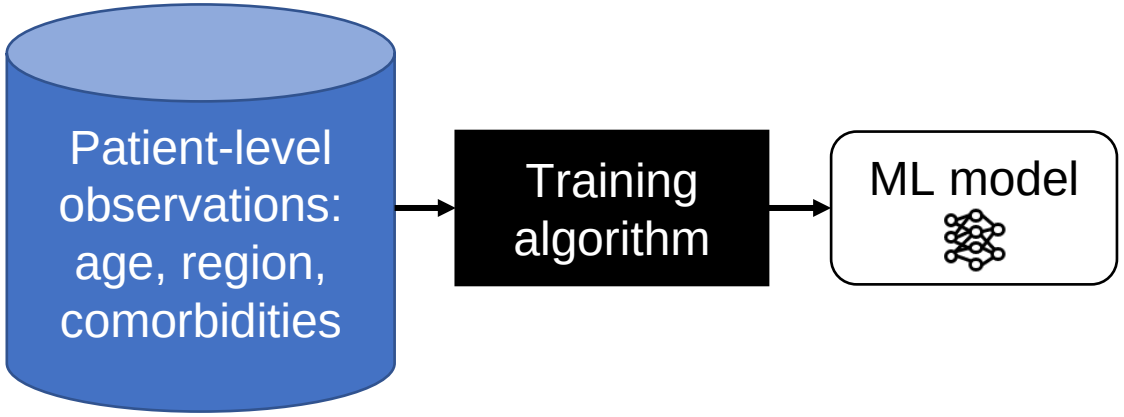


2. The company makes this model available as a service for users to query

3. The user makes a query: "Given Bob's profile (photos, posts, metadata), what pet does Bob have?"

Model sharing

1. A hospital trains a model to identify if a patient is at risk of COVID-19 complications based on their medical information.



2. The hospital shares this model (i.e., its weights) to third parties.

Training data privacy: General intuitions

- Although we want models to learn general “rules” for making predictions, they are always trained on specific, finite datasets.
- Because of how they are trained, models behave differently when applied to a training sample compared to an unseen sample.
- Privacy attacks are about characterizing and exploiting[1] this behavior to infer information about the training dataset.

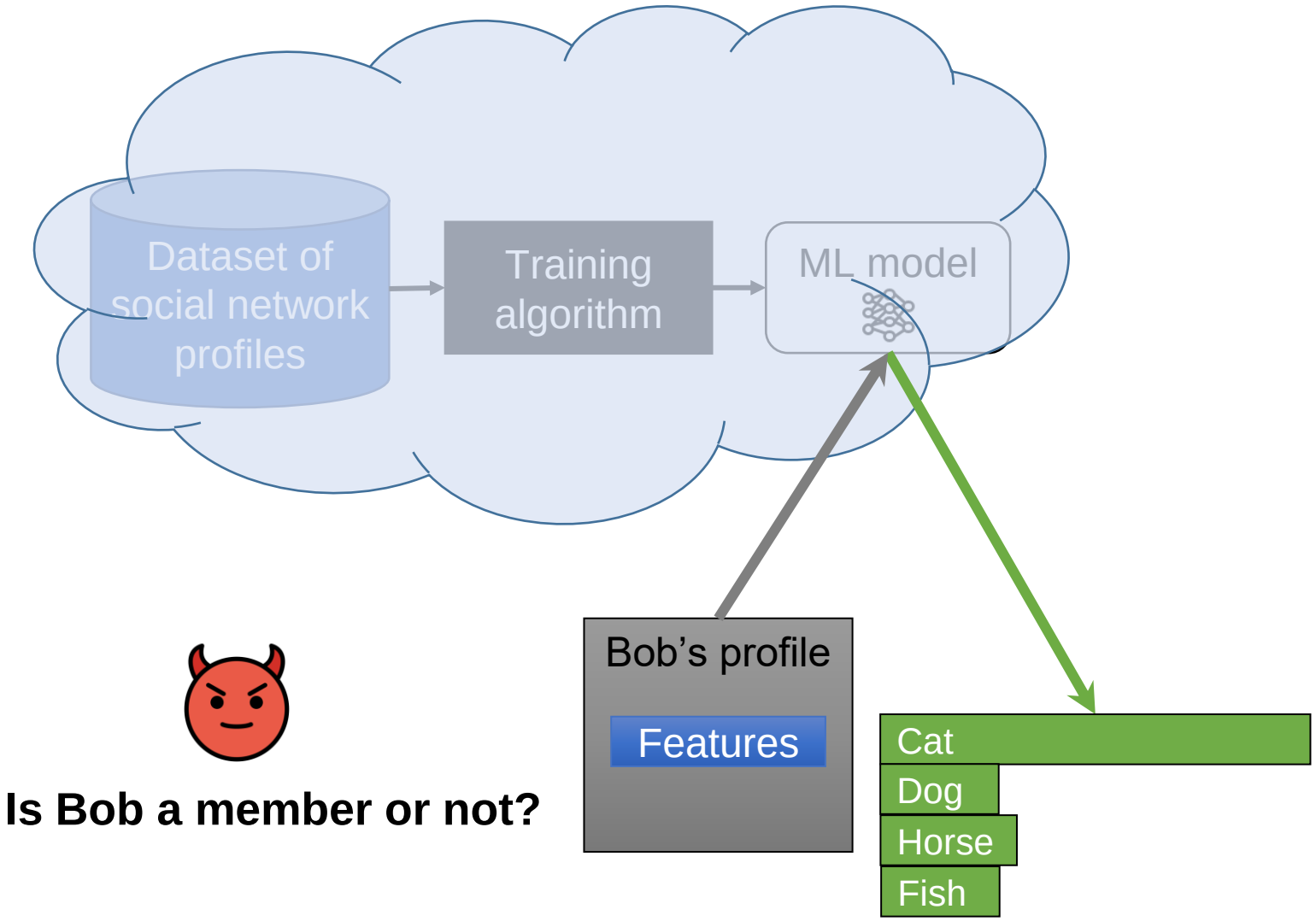
[1]Or inducing them in the active security model.

Membership inference attack

Formal definition of an MIA

- An adversary has access to:
  - A target model M_T trained on a private dataset $D \subset D^*$.
  - A target record $z_T = (x_T, y_T)$, e.g., $(Bob's\ profile, cat)$.
- The adversary wants to know if M_T was trained on (x_T, y_T), i.e., whether $(x_T, y_T) \in D$.
- Member record: any $(x, y) \in D$.
- Non-member record: any $(x, y) \in D^* \setminus D$.

Black-box membership inference

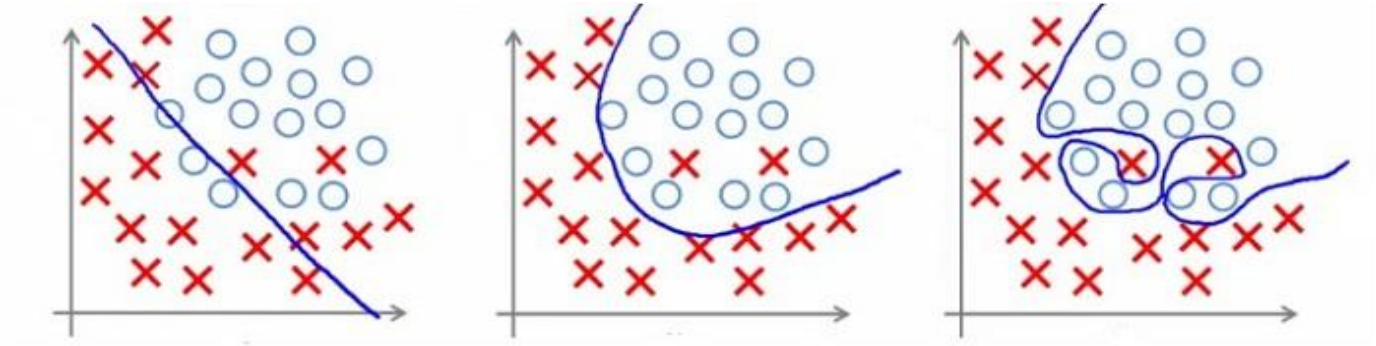


Black-box membership inference: Intuition

Ideally, we would like member and non-member samples to have the same confidence distribution:

However, the reality is:

Overfitting



How to construct an MIA

- We want to exploit that a model predicts the label of member records **more confidently** than the label of non-member records.
- This amounts to learning to distinguish between two distributions: the model confidences for members vs non-members.

Option 1

- The attacker can train a classifier A to distinguish between model confidences of member and non-members.
- The attacker can apply this classifier to the target record to obtain the membership prediction $A(x, y)$
- Strong adversary assumptions:
  - The adversary needs to know $D_{member} \subset D$ and $D_{nonmember} \subset D^* \setminus D$. This is needed to train the classifier on $\{(M_T(z), 1): z = (x, y) \in D_{member}\} \cup \{(M_T(z), 0): z = (x, y) \in D_{nonmember}\}$.
  - The adversary can query the model $|D_{member}| + |D_{nonmember}| + 1$ times.

Nasr, M., Shokri, R., & Houmansadr, A. (2019, May). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE symposium on security and privacy (SP) (pp. 739-753). IEEE.

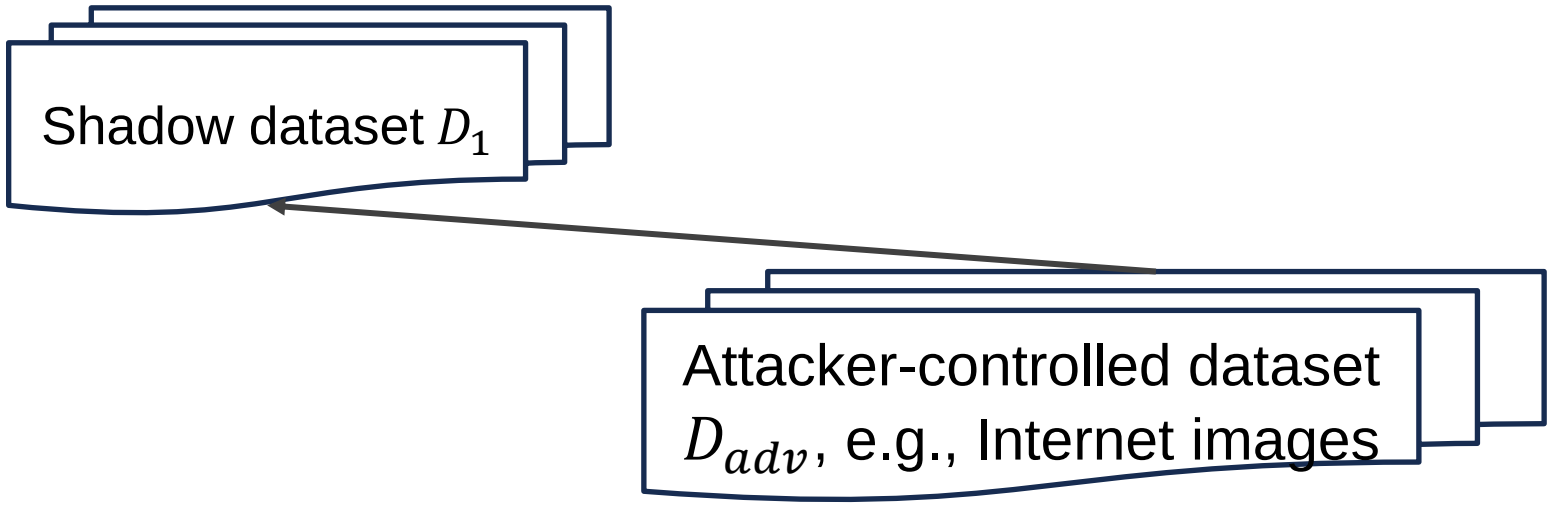
Option 2 – Shadow modeling technique

- Simulate the behavior of the model by training 💡 **shadow models** 💡
- The shadow models are trained using the same training algorithm T but on different datasets which are controlled by the attacker.
  - $M_1 = T(D_1, seed_1), \dots, M_k = T(D_K, seed_K)$ with corresponding member/non-member datasets $(D_1, D_{adv} \setminus D_1), \dots, (D_K, D_{adv} \setminus D_K)$.
  - The adversary then trains a classifier to distinguish between members and non-members based on confidence outputs of shadow models.
- Weaker adversary assumptions:
  - The adversary has a dataset D_{adv} different from the one used to train the target model (typically, from the same distribution)
  - The adversary can query the model once for a given target example x_T to obtain $y = M_T(x_T)$
  - The adversary knows the training algorithm T

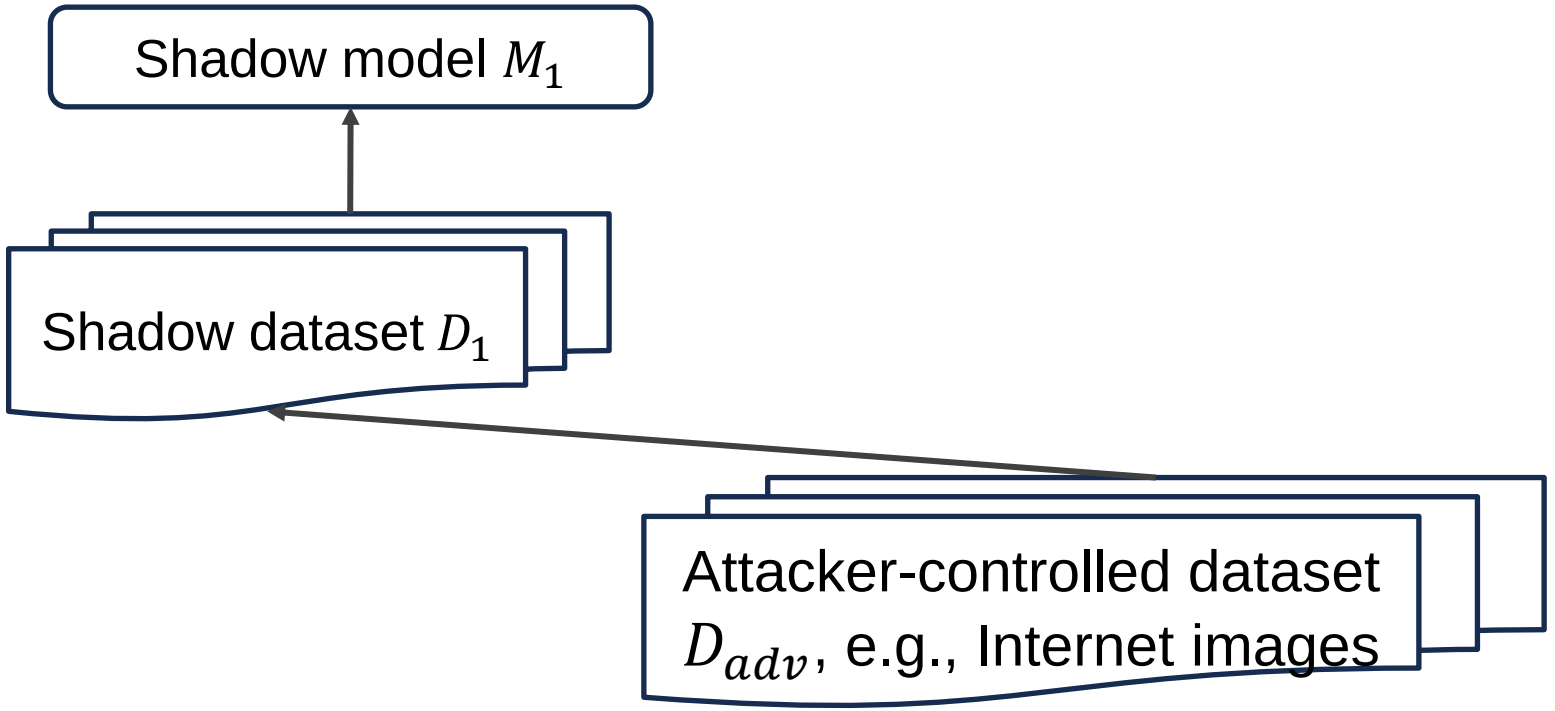
Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017, May). Membership inference attacks against machine learning models. In 2017 IEEE symposium on security and privacy (SP) (pp. 3-18). IEEE.

Shadow modeling idea: illustration

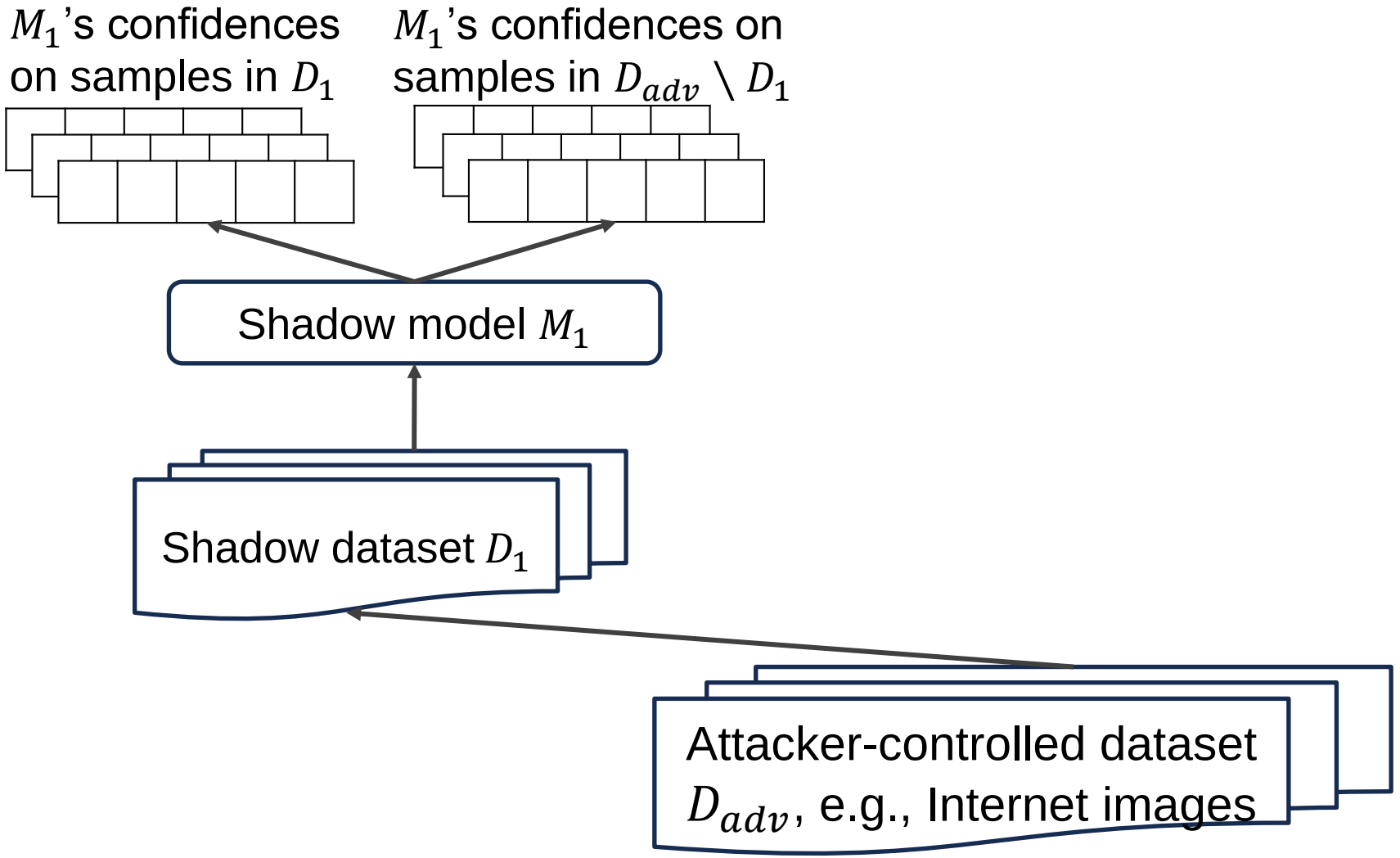
Shadow modeling idea: illustration



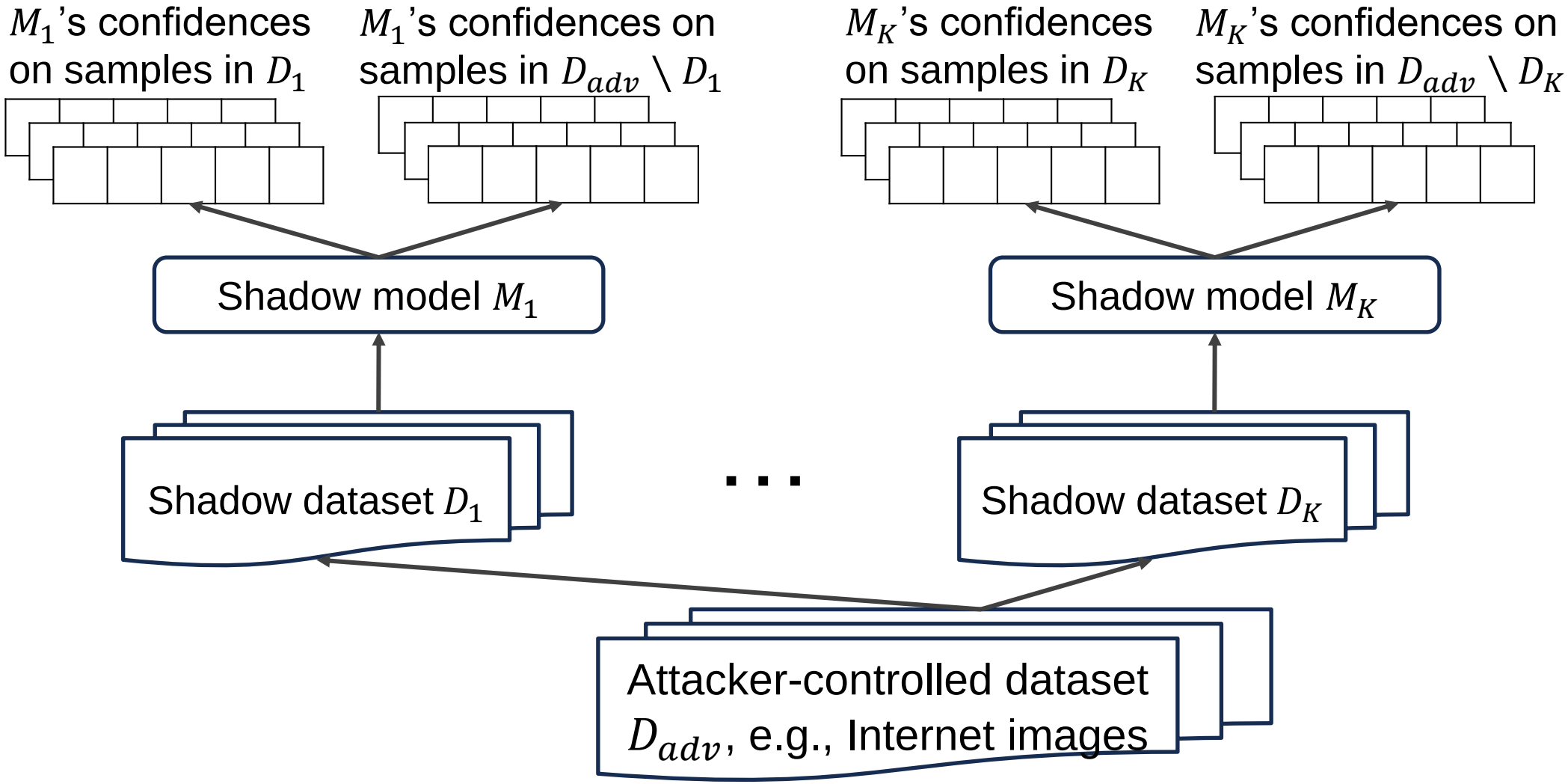
Shadow modeling idea: illustration



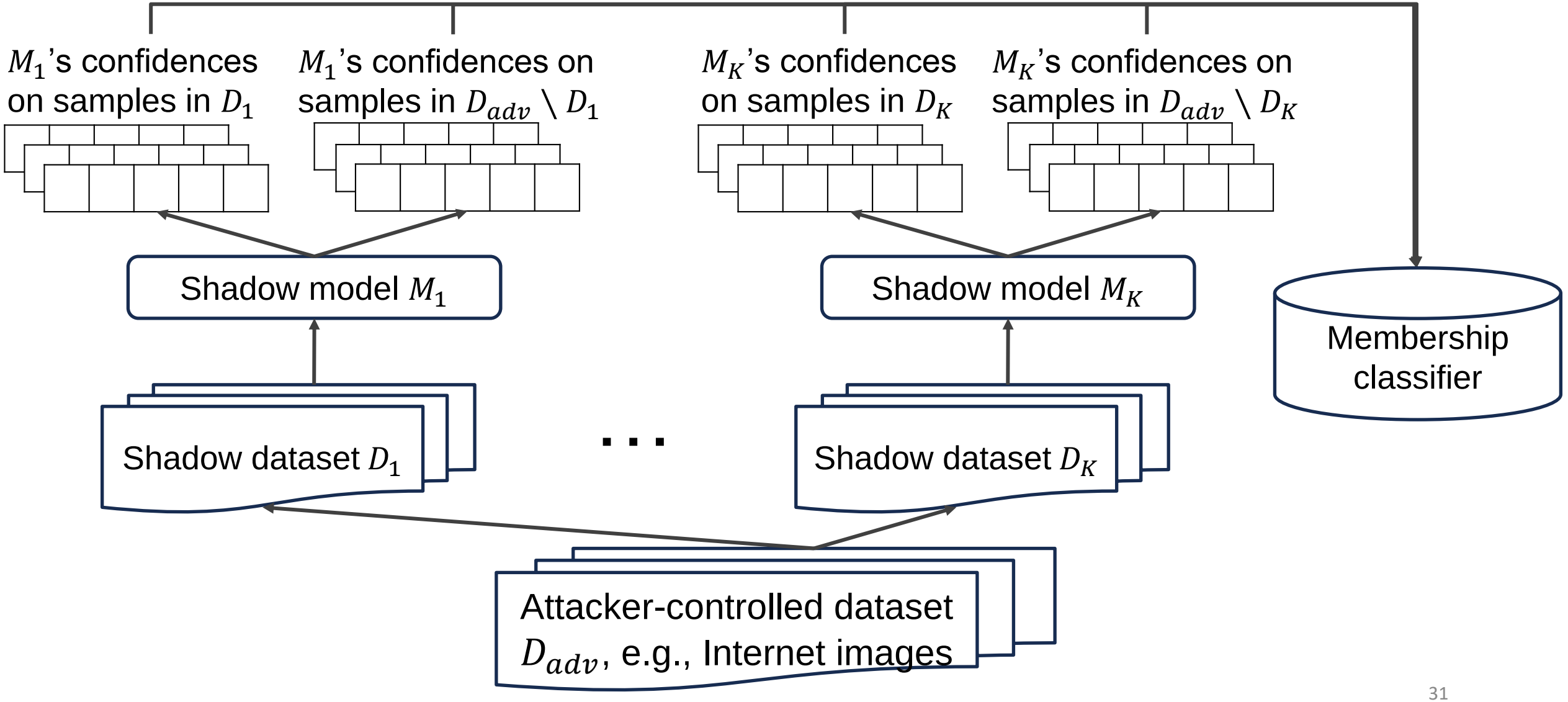
Shadow modeling idea: illustration



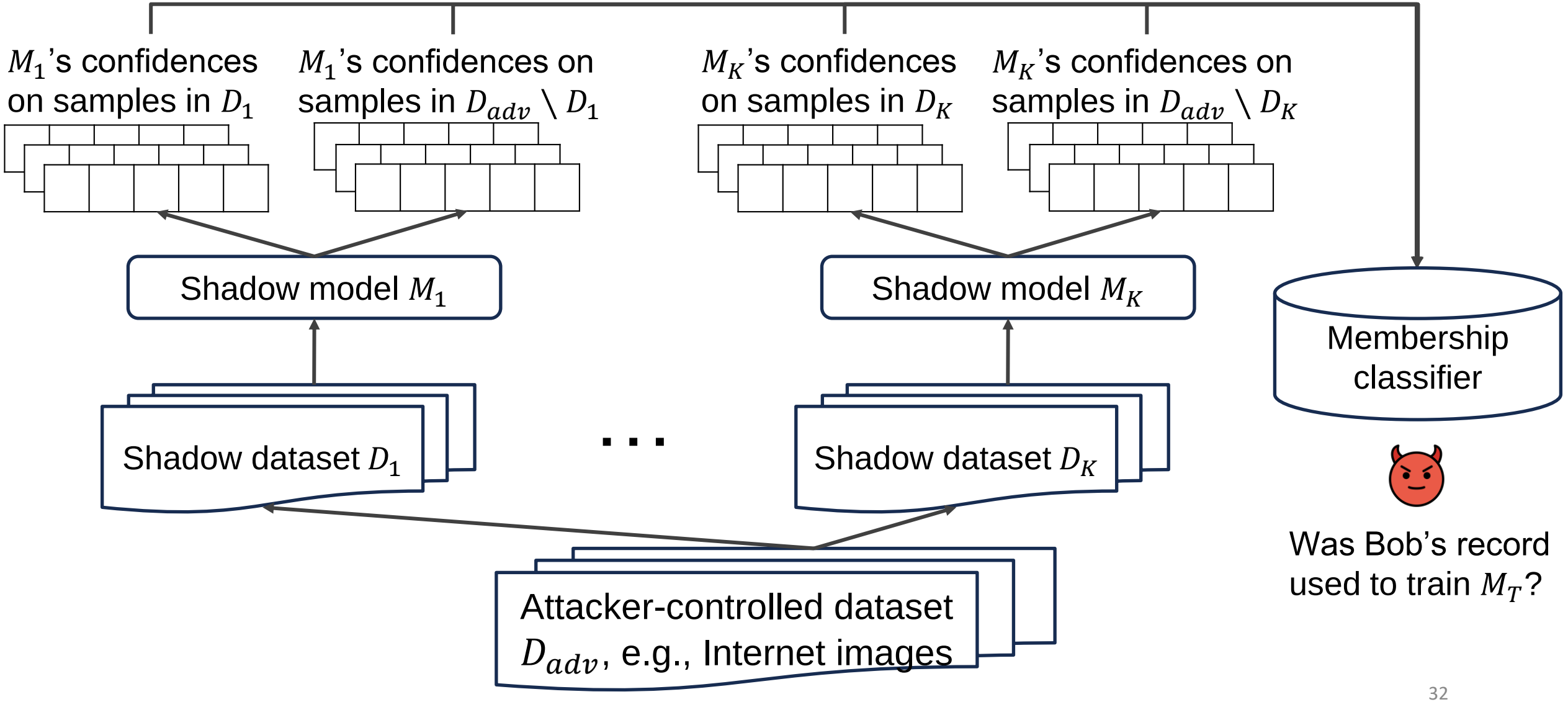
Shadow modeling idea: illustration



Shadow modeling idea: illustration



Shadow modeling idea: illustration



Option 3: Threshold attack

- A simple attack with interesting theoretical insights.
- Even weaker adversary assumptions:
  - No knowledge of the training algorithm and no access to the data distribution
  - The only knowledge about the target model:
    - The loss function $L(\cdot,\cdot)$ used to train the model.
    - The average training loss $L^* = \frac{1}{n}\sum_{i=1}^{n} L(M_T(x_i), y_i)$
  - The adversary can query the model once for a given target example x_T to obtain $y = M_T(x_T)$
- How to attack in this setting?

Relationship between model confidences and loss function

- **Model confidences**: Given x, the model's confidences on each class is a vector

$$M(x) = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_C),\ \sum_i \hat{y}_i = 1$$

- **At train time**, neural network's weights on the cross-entropy loss:

$$L(x, y) = -\log(\hat{y}_y)$$

where $\hat{y}_y$ is the model's confidence on x's true class, y

Threshold attack

ThresholdAttack$(M, (x, y), L^*, L(.))$:

1. Query $\hat{y} = M(x)$
2. Compute loss at target example $L = L(\hat{y}, y)$
3. Compare the target's loss with average loss

   if $L \leq L^*$: predict "member", i.e., that (x, y) was in the training dataset

   else: predict "non-member", i.e., that (x, y) was not in the training dataset

Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018, July). Privacy risk in machine learning: Analyzing the connection to overfitting. In 2018 IEEE 31st computer security foundations symposium (CSF) (pp. 268-282). IEEE.

Estimating vulnerability with the threshold attack

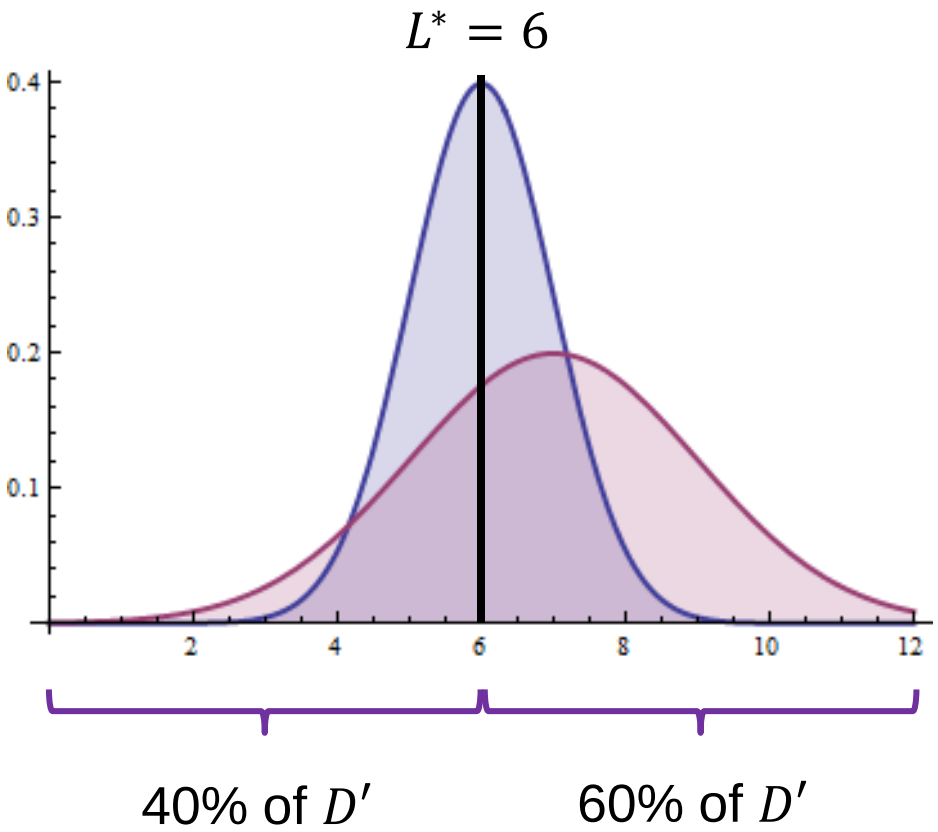
Consider:

- A training dataset D whose loss distribution is shown in **blue**
- A test dataset D' whose loss distribution is shown in **purple**

The adversary knows

- The average loss on the training set is $L^* = 6$
- Prior probability of a sample being a member is 0.5

How can we estimate the advantage ($Adv = P(success) - P(prior)$) of the threshold attack over random guessing on all examples from D and D'?



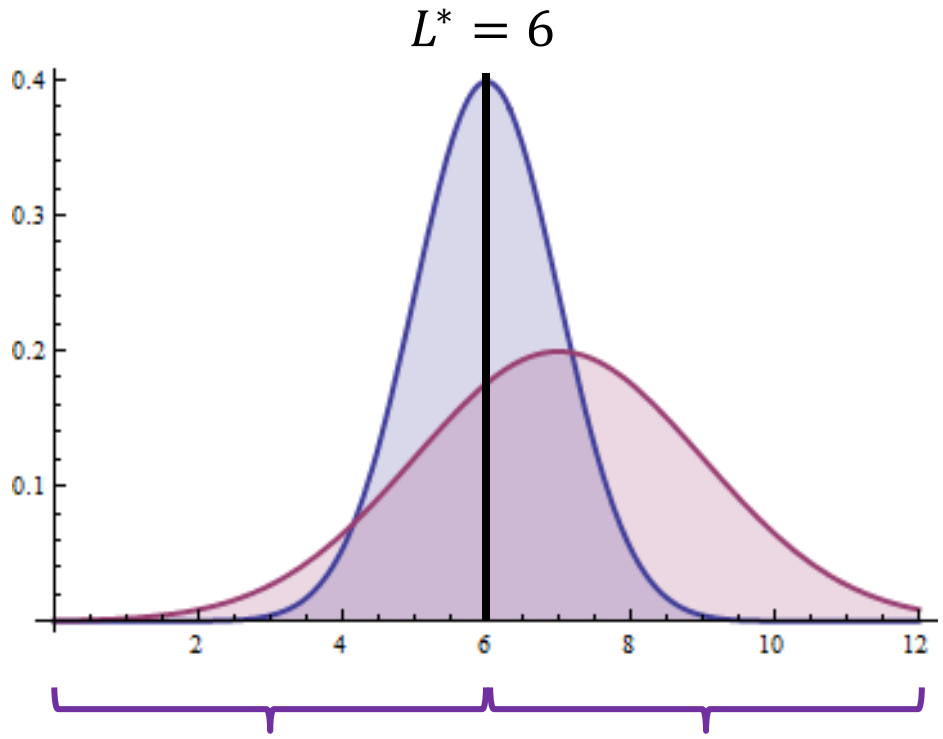
Estimating vulnerability with the threshold attack

How can we estimate the advantage ($Adv = P(success) - P(prior)$) of the threshold attack over random guessing on all examples from D and D'?

- A: adversary guess, M: membership of sample
- $P(prior) = P(M = 1) = 0.5$
- $P(success) = P(A = M)$
  $= P(A = 1, M = 1) + P(A = 0, M = 0)$
  $= P(A = 1|M = 1) \times P(M = 1) +$
  $P(A = 0|M = 0) \times P(M = 0)$
  $= 0.5 * 0.5 + 0.6 * 0.5 = 0.25 + 0.3 = 0.55.$
- $Adv = 0.55 - 0.5 = 0.05$
- Adv can range between 0 and 0.5. We can multiply it by 2 to have it range between 0 and 1.

$$Adv = 2P(A = M) - 1 = P(A = 1|M = 1) - P(A = 1|M = 0)$$

(adversary's TPR minus adversary's FPR)



Relationship to overfitting

Theorem (informal statement, Yeom et al., 2018): Assume the loss function is bounded by a constant B. For a sample (x, y), the adversary who outputs 1 (member) with probability $L(\hat{y}, y)/B$ and 0 otherwise has an advantage equal to R_{gen}/B.

Average generalization error of training algorithm T:

$$R_{gen}(T, n, L) = E_{D\sim(X,Y)^n,(x,y)\sim(X,Y)}[L(T(D)(x), y)] - E_{D\sim(X,Y)^n,(x,y)\sim D}[L(T(D)(x), y)]$$

Expected loss over samples (x, y) of the distribution (X, Y)

Expected loss over samples (x, y) of the training dataset D

Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018, July). Privacy risk in machine learning: Analyzing the connection to overfitting. In 2018 IEEE 31st computer security foundations symposium (CSF) (pp. 268-282). IEEE.

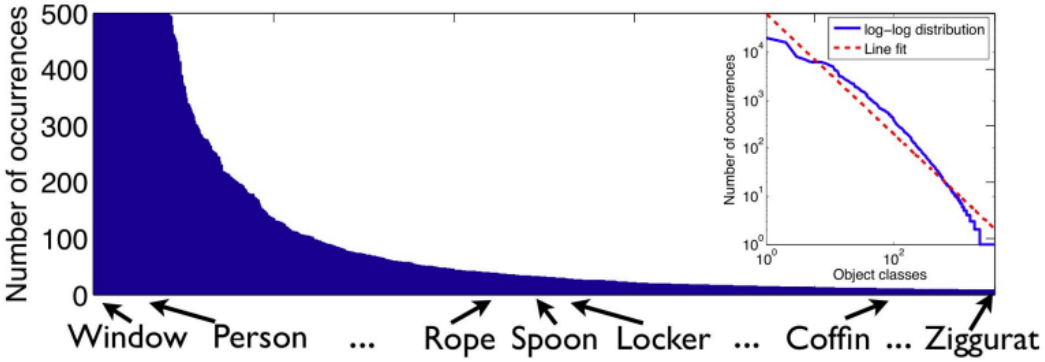
Lower bound on vulnerability

- Overfitting implies vulnerability to membership inference attacks.

- Practical estimation of the lower bound on vulnerability:
  - The threshold attack is extremely simple and cheap: adversarial advantage is the difference between proportions of examples under loss threshold on training/test datasets.
  - Surprisingly, it is also rather powerful in practice.

Is overfitting necessary to obtain good classifiers?

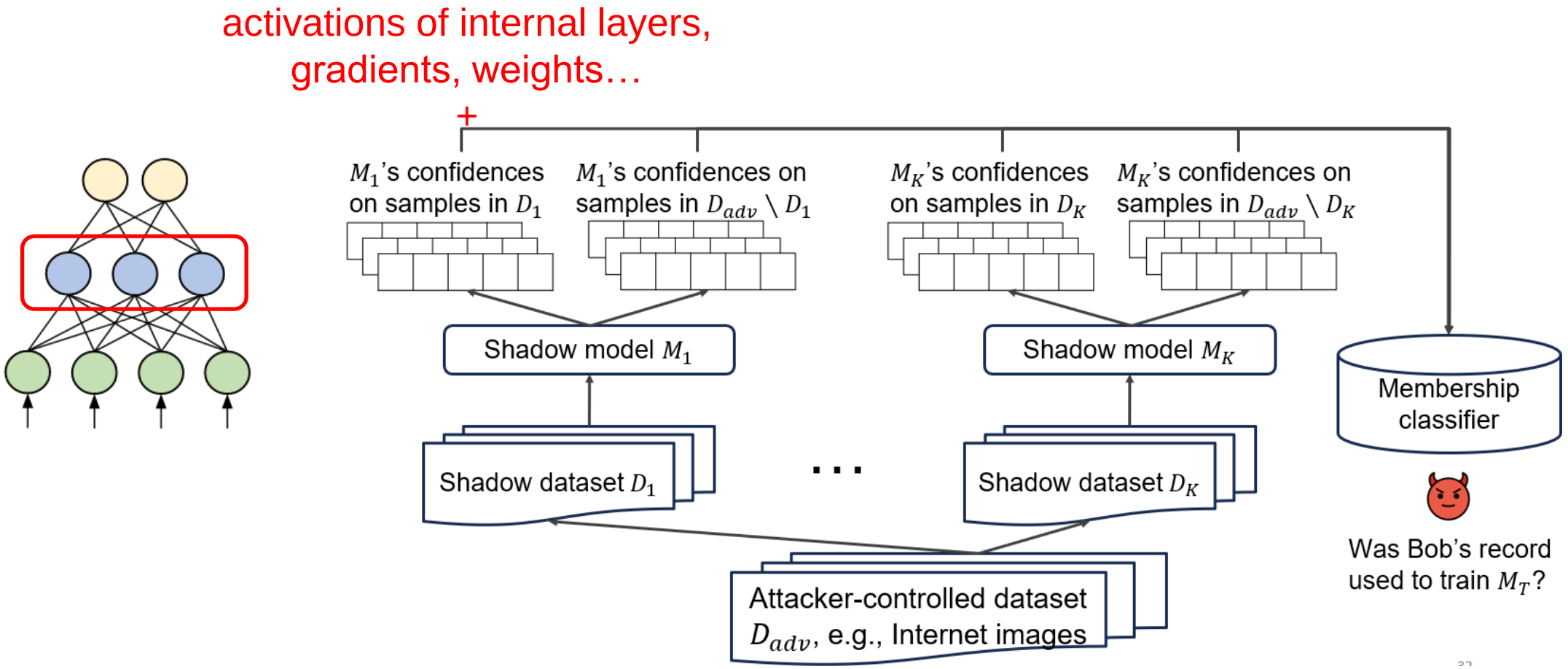
- Long-tailed data distribution: A distribution in which the frequencies of sub-populations are long-tailed, i.e., there are many sub-populations with small frequencies.
  - A dataset sampled from a long-tailed distribution will have few samples available for every sub-populations.
  - Real-world examples: images, text,…
- **Theorem [Feldman, 2019, informal statement]:** To achieve optimal performance, a model trained on long-tailed data must memorize the labels of sub-population samples in the training dataset.



(a) The number of examples by object class in SUN dataset

Feldman, V. (2020, June). Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (pp. 954-959).

White-box shadow modeling attacks

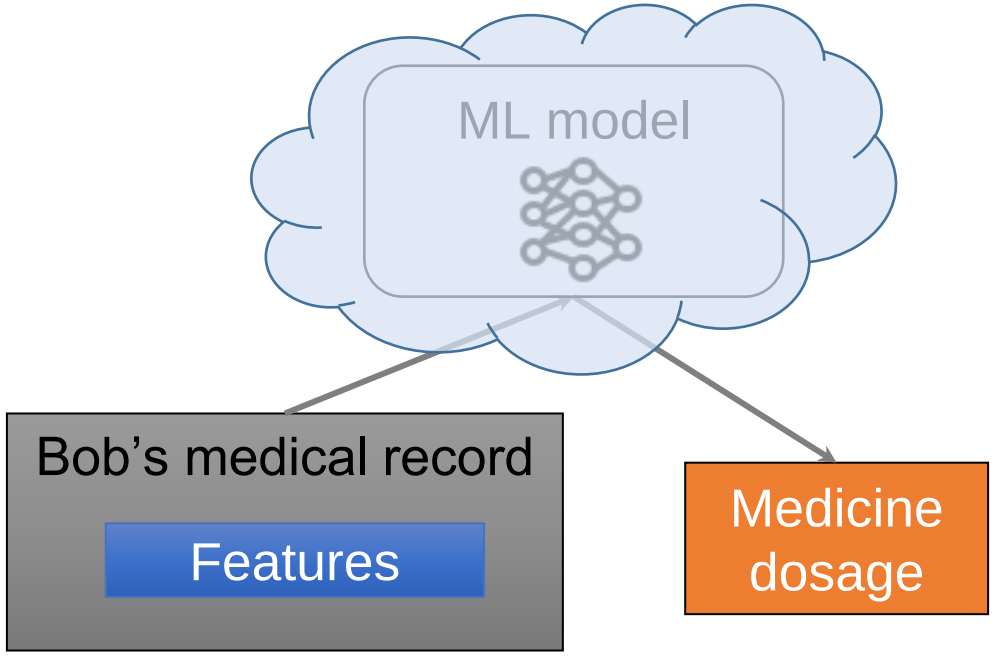


Attribute inference attack

- Threat model
  - The attacker knows the partial record of an individual Bob. I.e., if Bob's record has n attributes $x_1, \ldots, x_n$ and label y, the attacker knows $x' = (x_1, \ldots, x_{n-1}, y)$.
  - The adversary's goal is to infer the missing attribute x_n given access to a target model trained on (x, y).

Connection between attribute inference and membership inference

- Imagine an ML model that takes medical information and returns a recommended dosage of some medicine.
- Assume we have an AIA for inferring the “Has disease” attribute. How can we use it to construct an MIA?
  - Use the AIA to query the model on Bob’s value for “Has disease” given x'. If the attribute retrieved is correct (i.e., the AIA returns “yes”), predict “member”. Else, predict “non-member”.
  - Yeom et al. showed this MIA to have the same advantage as the AIA.
- A reduction exists in the opposite direction, but the advantage of the MIA-based AIA is smaller than the MIA’s.



Sex	Age	Country	Has disease
M	55	Belgium	yes

Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018, July). Privacy risk in machine learning: Analyzing the connection to overfitting. In 2018 IEEE 31st computer security foundations symposium (CSF) (pp. 268-282). IEEE.

Summary

-  The training process makes the model behave differently on training data.
-  An attacker can exploit this behavior to infer information about the training data.

Defending against MIAs using differential privacy

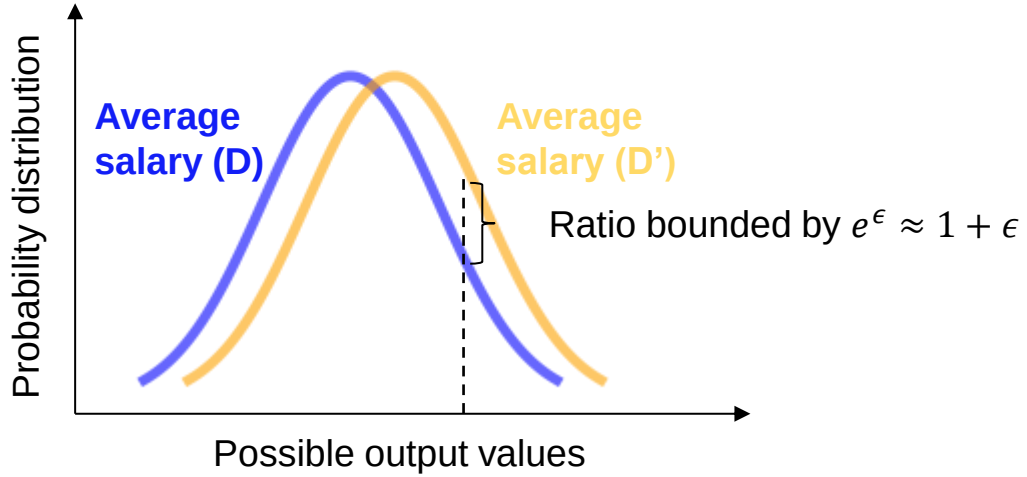
- Differential privacy (DP) is a mathematical framework to reason rigorously about privacy.
- So far, you've studied DP in the context of statistical queries.
- The same ideas apply to ML algorithms, too!
  - With some differences we'll explore today
- Key intuition: The outputs of the training algorithm should not "change too much" regardless of the inclusion or exclusion of any one record

Differential privacy

Definition (ϵ-Differential Privacy): Let $\epsilon > 0$. A randomized mechanism T is said to satisfy ϵ-differential privacy (or to be ϵ-DP) if for every pair of neighboring datasets D, D' (i.e., differing in exactly one record) and for any subset of outputs Y, the following holds:

$$\Pr(T(D) \in Y) \leq e^{\epsilon} \Pr(T(D') \in Y)$$

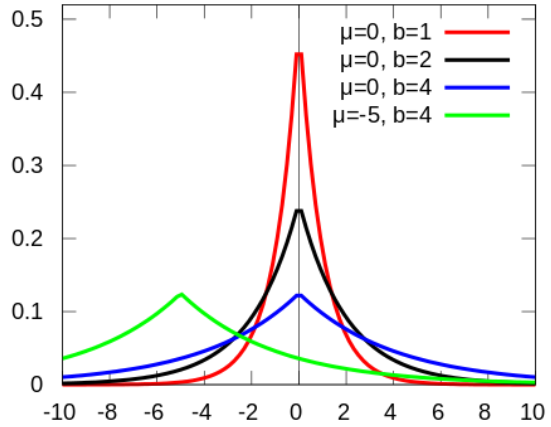
where the randomness is taken over the coin flips of Y.



Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006.

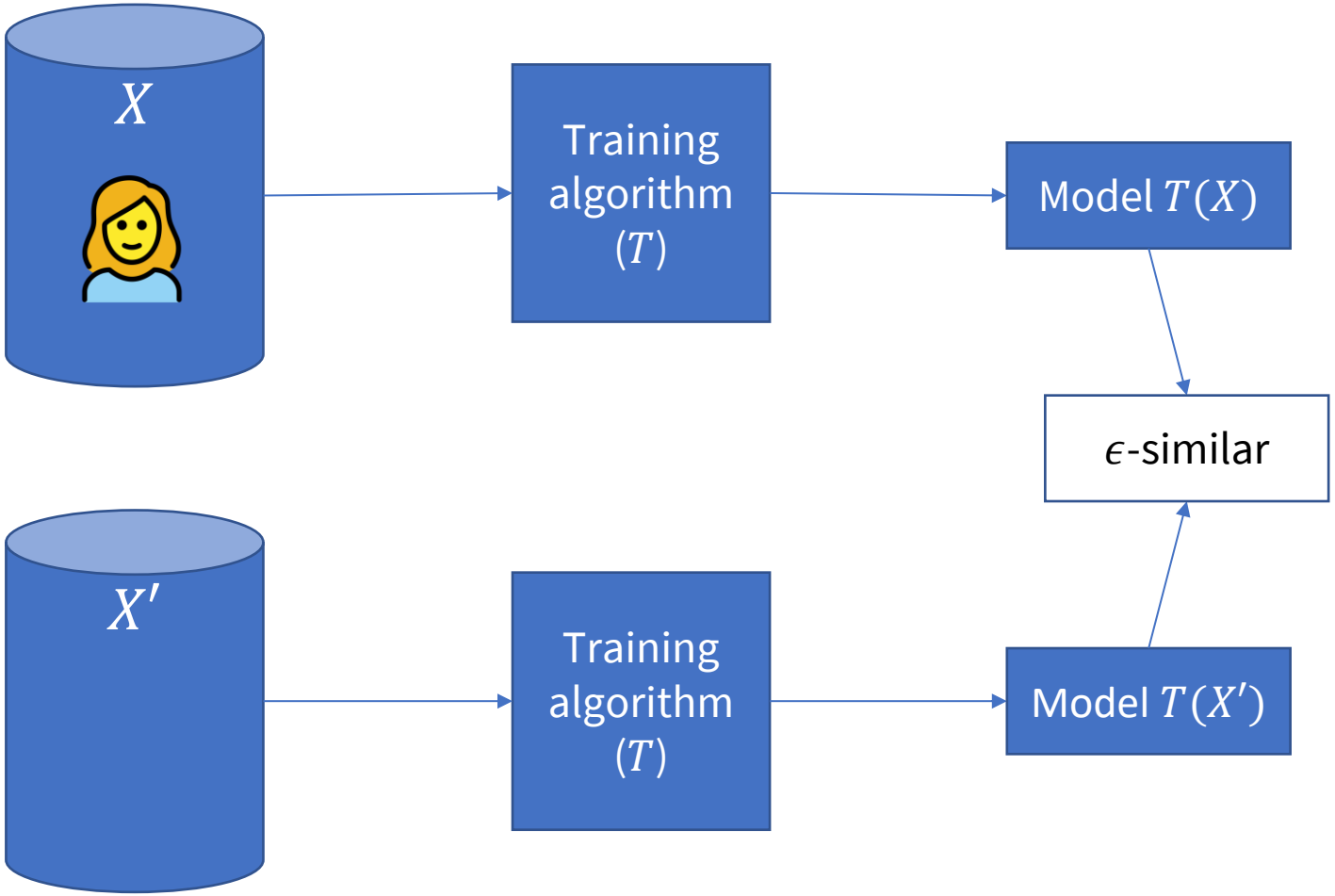
Achieving DP through noise addition

- Noise addition is one of the key ingredients of DP mechanisms.
- Global sensitivity of the mechanism $\Delta T = \max_{D \sim D'} |T(D) - T(D')|$
- The global sensitivity captures how much a single individual's data can change the answer to the query in the worst case, and therefore, the uncertainty in the response that we must introduce to hide anyone's participation.
- Theorem: Given a mechanism T, adding noise $Lap\left(0, \frac{\Delta T}{\epsilon}\right)$ to a query ensures ϵ-DP.



Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006.

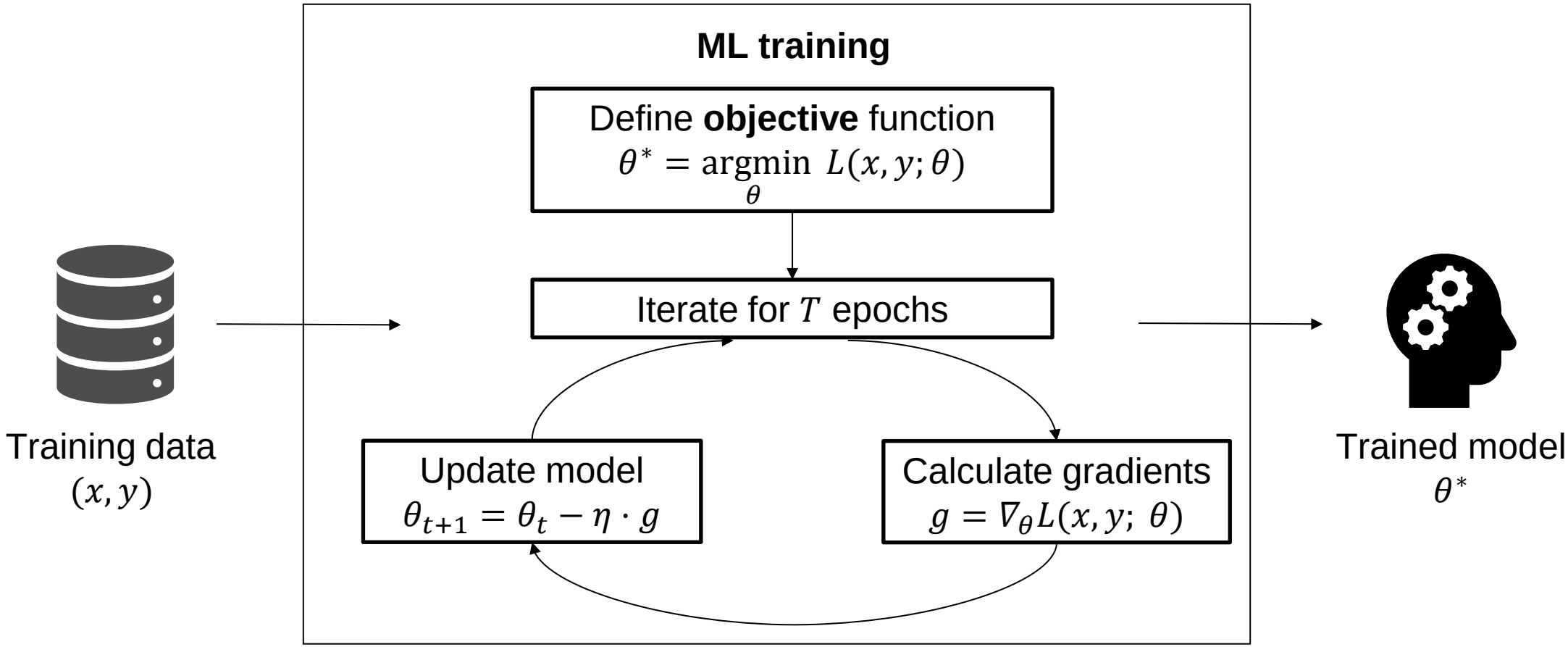
Differentially private ML



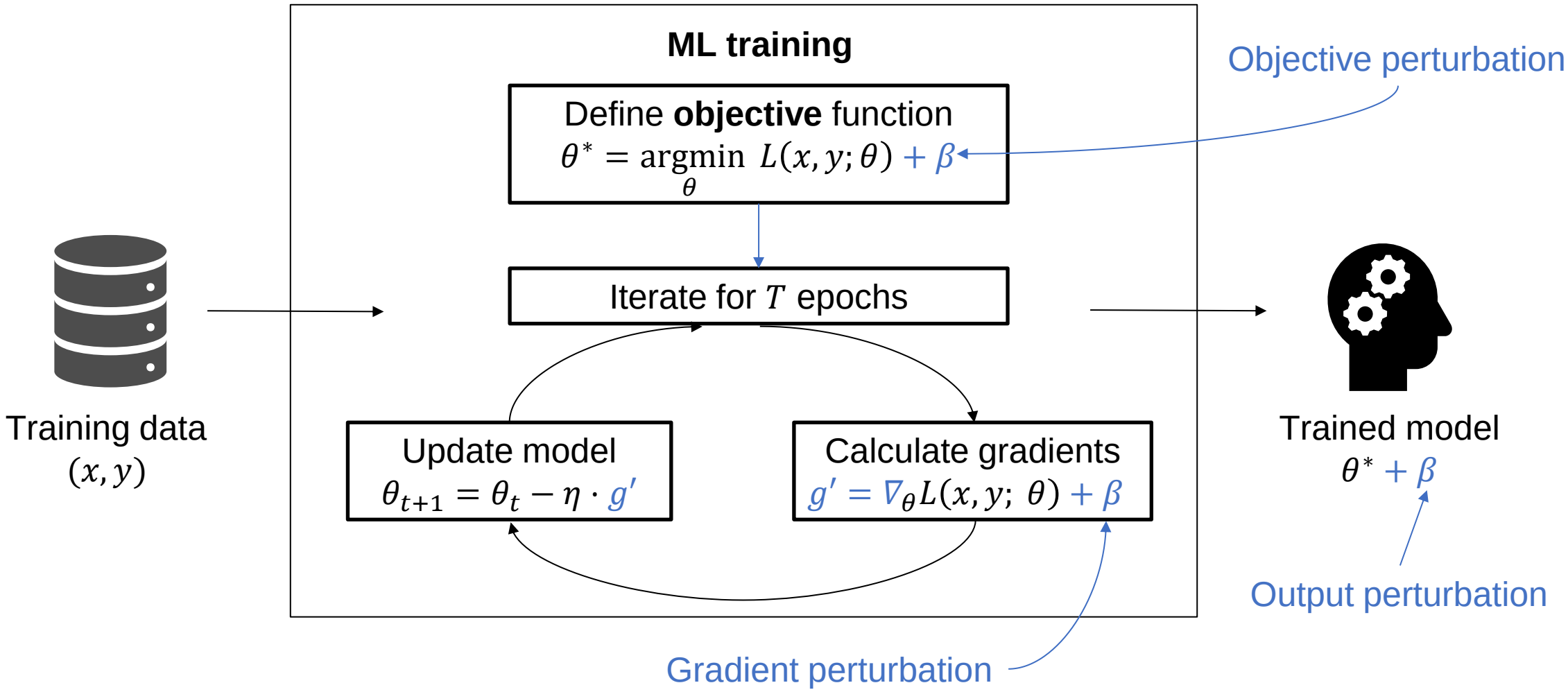
$$\frac{P(T(X) = M)}{P(T(X') = M)} = e^{\varepsilon} \approx 1 + \varepsilon$$

Differentially private ML

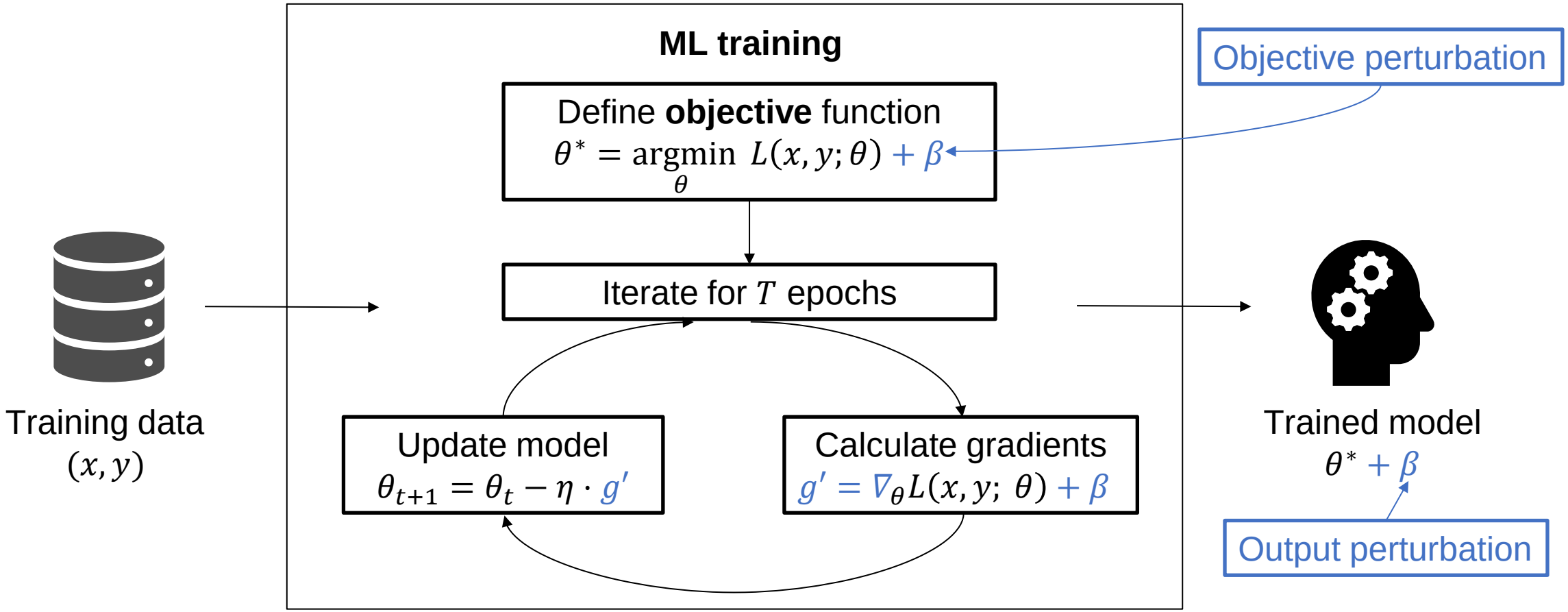
Loss minimisation



Differentially private ML
Techniques

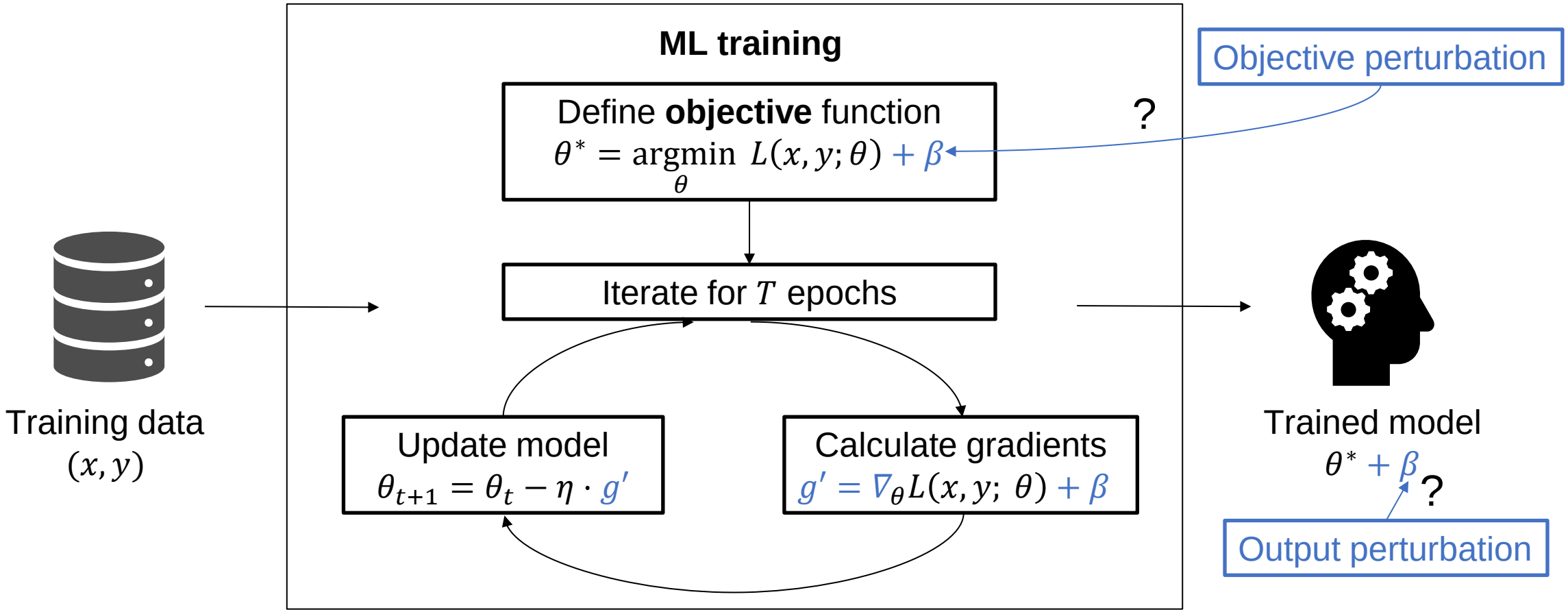


Differentially private ML
Techniques



How much can a single training record affect each of these functions?

Differentially private ML
Techniques

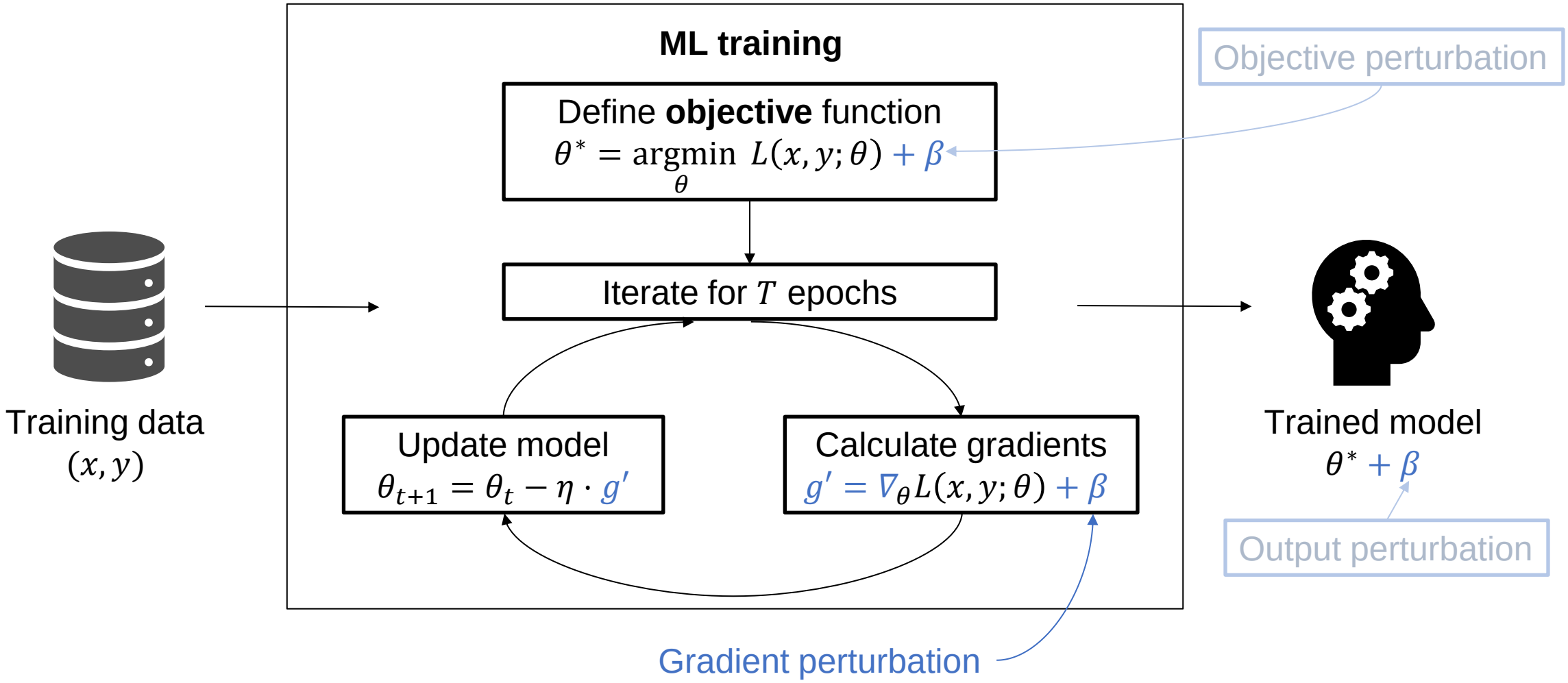


How much can a single training record affect each of these functions?

We do not know how to calculate a bound on the sensitivity

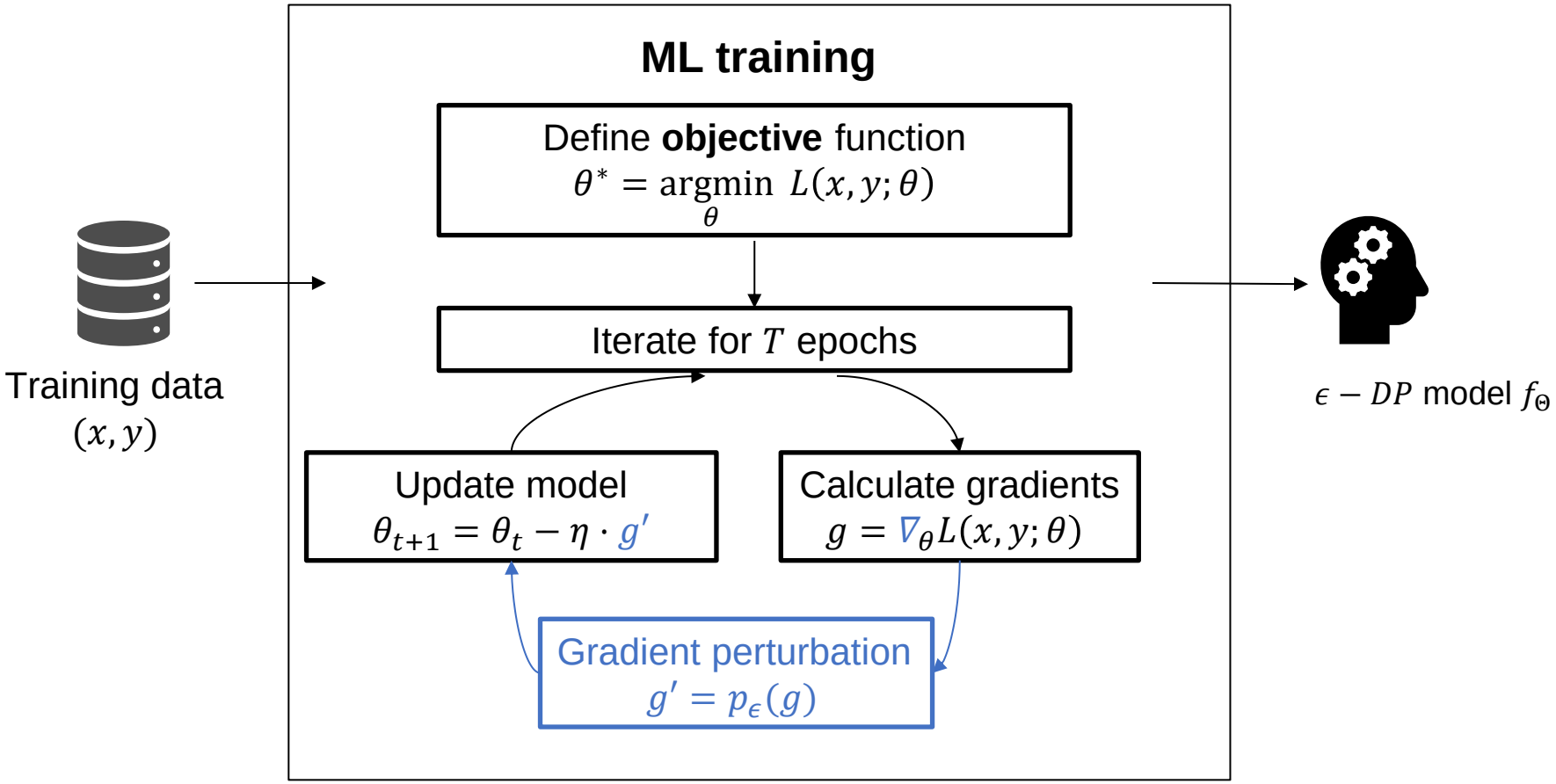
Differentially private ML

Techniques



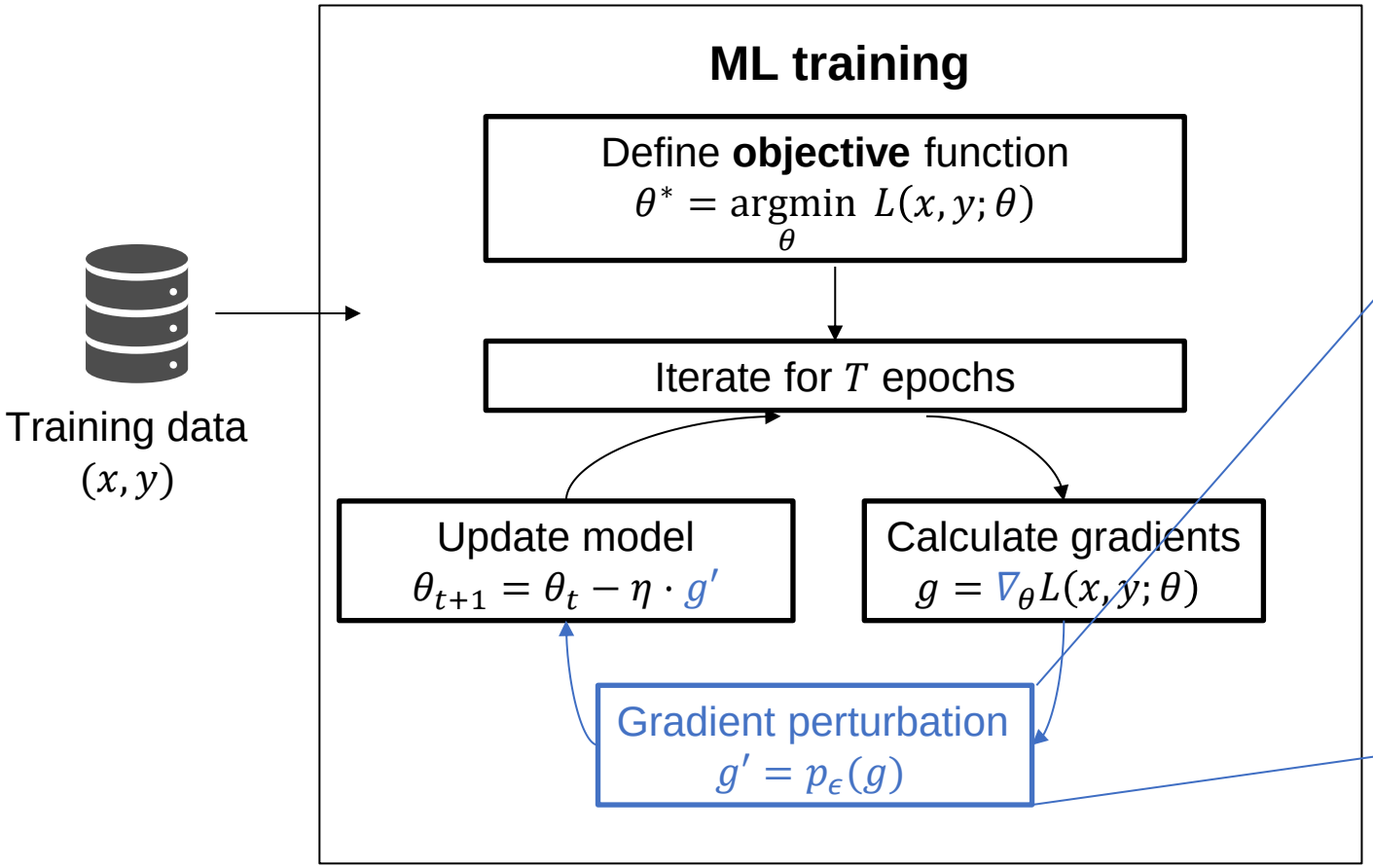
Differentially private ML

Differentially private stochastic gradient descent (DP-SGD)



Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 308-318).

Differentially private ML
DP-SGD



Gradient perturbation:

1. Gradient clipping (bound sensitivity):
$$g_C = g \times min(\frac{C}{||g||_2}, 1)$$
2. Noise sampling based on sensitivity and ϵ:
$$d = N(0, \sigma^2 C^2)$$
3. Noise application:
$$g' = g_C + d$$

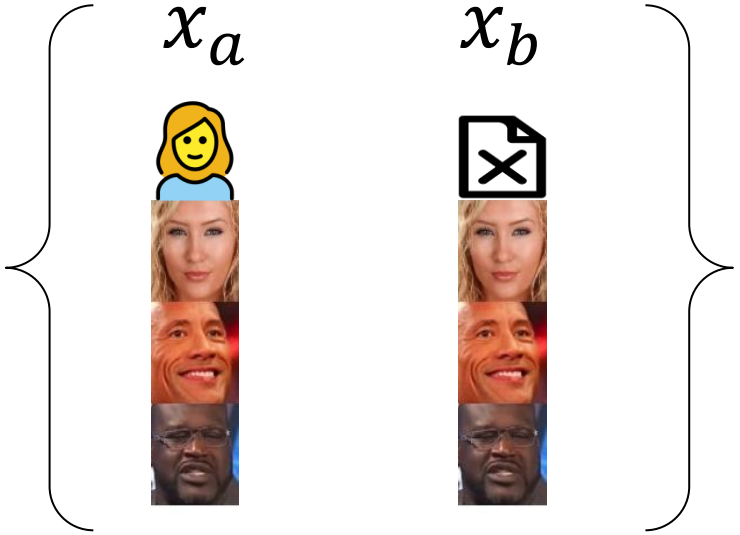
DP parameters:
C is the imposed sensitivity
σ is a function of ϵ
N is the isotropic gaussian dist.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 308-318).

A bound to gradient sensitivity:

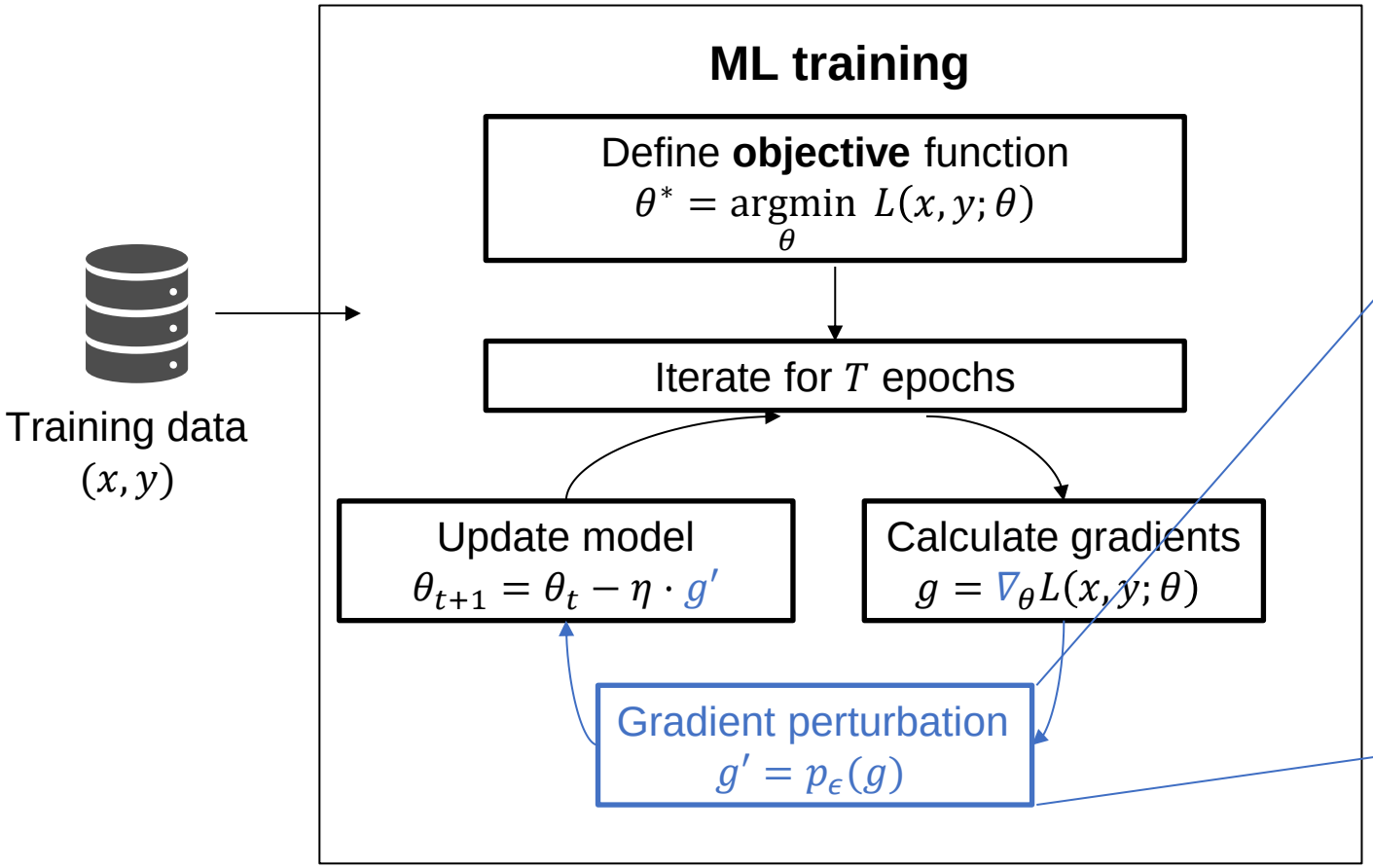
- The gradient does not have a fixed range (i.e., it ranges between $[-\infty, +\infty]$).
- What we need is:

For every pair of neighbor batches:



$$\left|\left|\sum_{x \in x_a} \nabla_\theta L(M_\theta(x), y) - \sum_{x \in x_b} \nabla_\theta L(M_\theta(x), y)\right|\right|_2 \leq C$$

Differentially private ML
DP-SGD

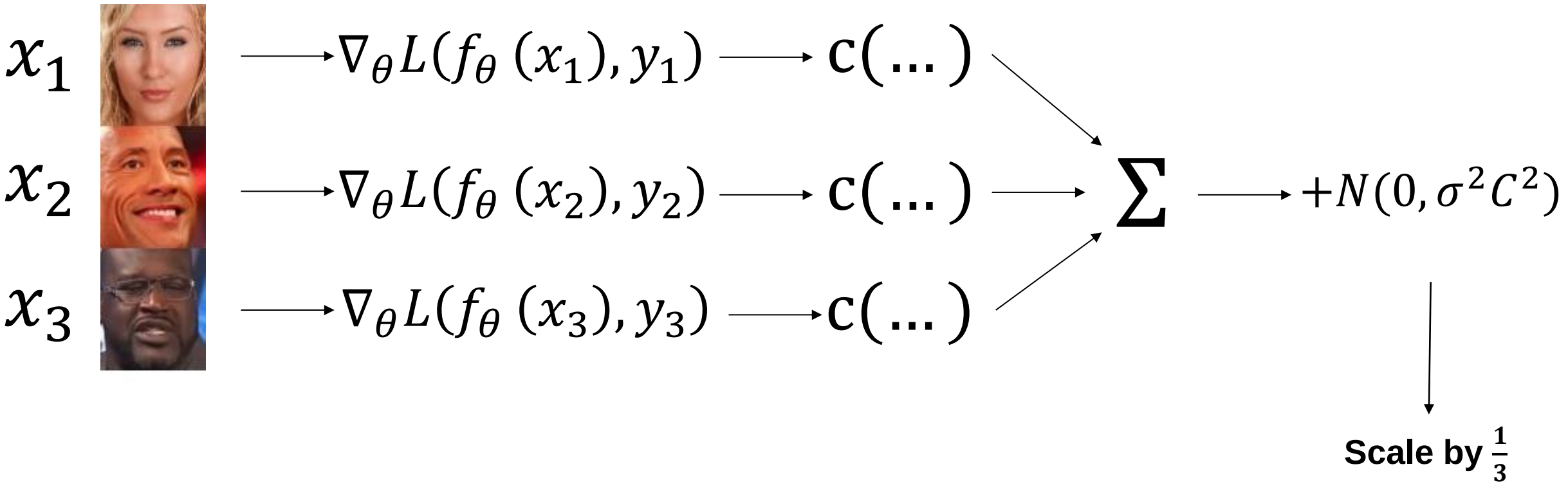


Gradient perturbation:

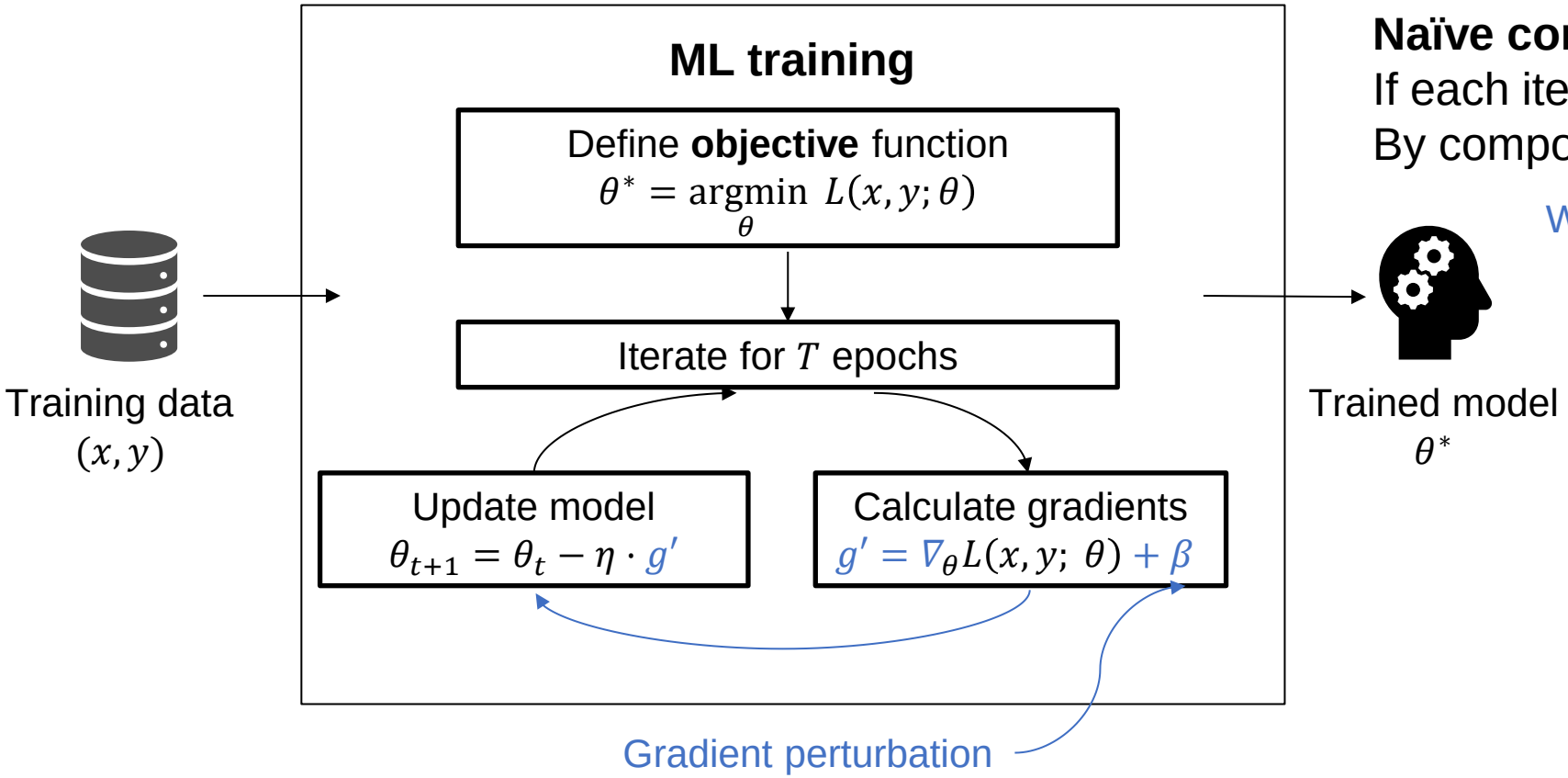
1. Gradient clipping (bound sensitivity):
$$g_C = g \times min(\frac{C}{||g||_2}, 1)$$
2. Noise sampling based on sensitivity and ϵ:
$$d = N(0, \sigma^2 C^2)$$
3. Noise application:
$$g' = g_C + d$$

DP parameters:
C is the imposed sensitivity
σ is a function of ϵ
N is the gaussian distribution

DP-SGD in details



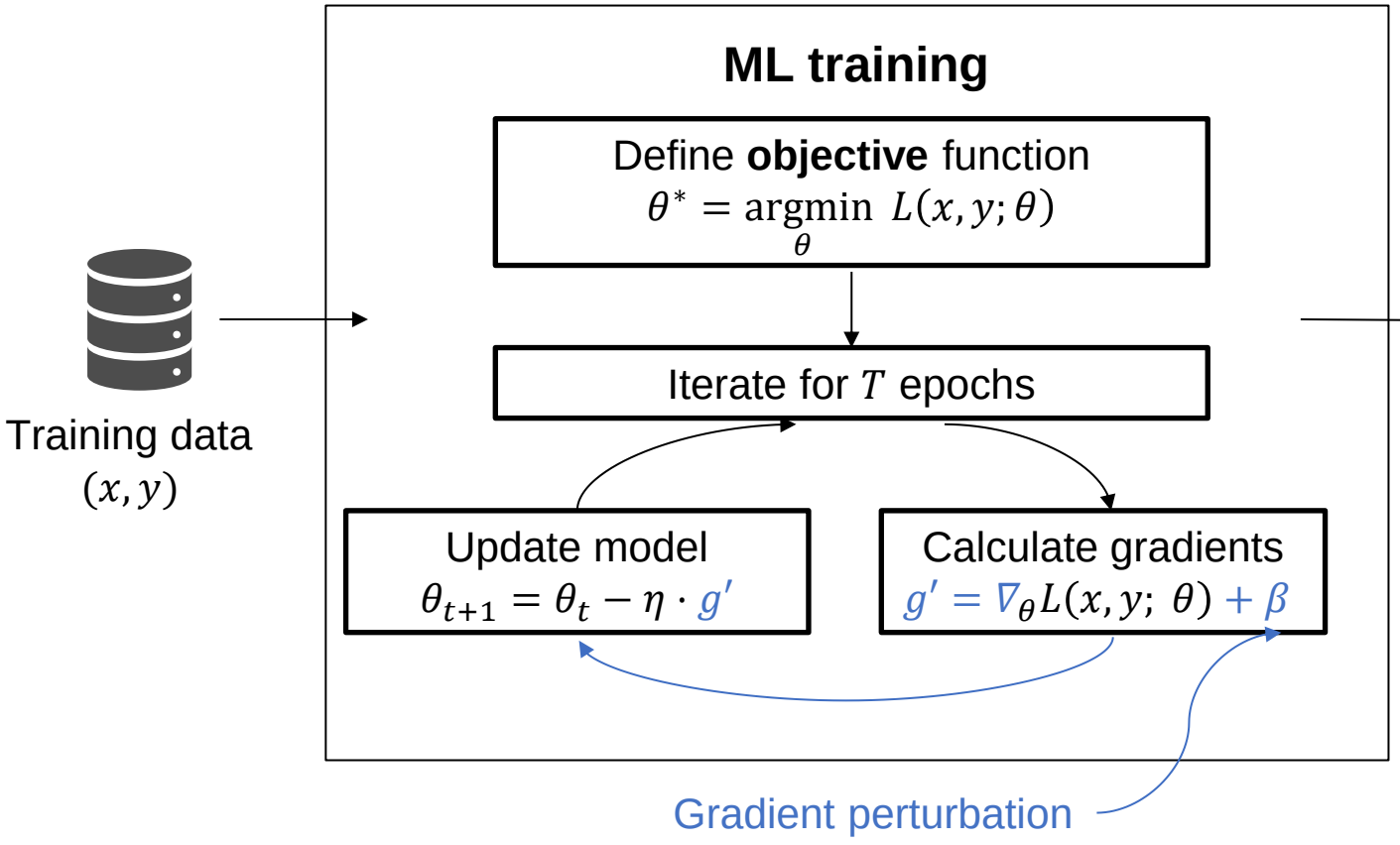
Differentially private ML
DP-SGD



Naïve composition:
If each iteration is $\boldsymbol{\varepsilon}$ –**DP**
By composition, final model is $\mathbf{T}\boldsymbol{\varepsilon} - \boldsymbol{DP}$

We end up consuming too much budget

Differentially private ML
DP-SGD



Naïve composition:
If each iteration is $\boldsymbol{\varepsilon}$ –**DP**
By composition, final model is $\boldsymbol{T\varepsilon - DP}$

We end up consuming too much budget

Advanced composition:
If each iteration is $\boldsymbol{\varepsilon}$ –**DP**
By **advanced** composition, final model is
$\boldsymbol{(\mathcal{O}(\sqrt{T}\varepsilon), \delta) - DP}$

We need to add less noise in each iteration
to get the same total epsilon at the end

Differentially private ML
Advanced composition

Naïve composition:
If each iteration is $\boldsymbol{\varepsilon}$ –**DP**
By composition, final model is
$\mathbf{T}\boldsymbol{\varepsilon} - \boldsymbol{DP}$

Advanced composition:
If each iteration is $\boldsymbol{\varepsilon}$ –**DP**
By **advanced** composition, final model is
$(\boldsymbol{\mathcal{O}}(\sqrt{\boldsymbol{T}}\boldsymbol{\varepsilon}), \boldsymbol{\delta}) - \boldsymbol{DP}$

We “gain” a factor $\sqrt{\boldsymbol{T}}$

Differentially private ML
Advanced composition

Naïve composition:
If each iteration is $\boldsymbol{\varepsilon}$ −**DP**
By composition, final model is
$\mathbf{T}\boldsymbol{\varepsilon} - \boldsymbol{DP}$

Advanced composition:
If each iteration is $\boldsymbol{\varepsilon}$ −**DP**
By **advanced** composition, final model is
$(\mathcal{O}(\sqrt{\boldsymbol{T}}\boldsymbol{\varepsilon}), \boldsymbol{\delta}) - \boldsymbol{DP}$

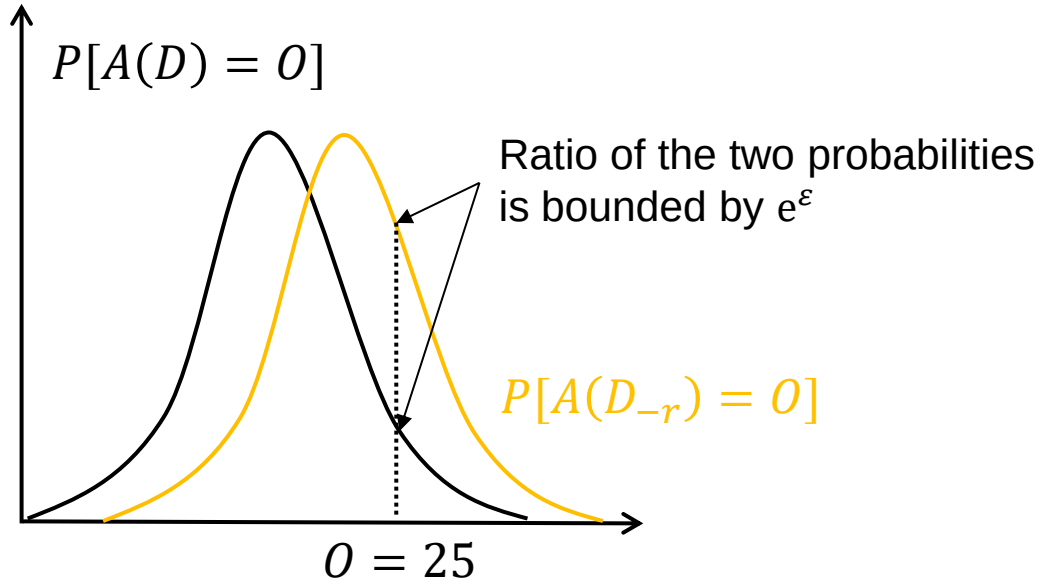
We "gain" a factor $\sqrt{\boldsymbol{T}}$

At the cost of introducing a failure rate

What is the $\boldsymbol{\delta}$ in (ε, δ) − DP?

Differentially private ML
Advanced composition

Naïve composition: $\varepsilon - DP$

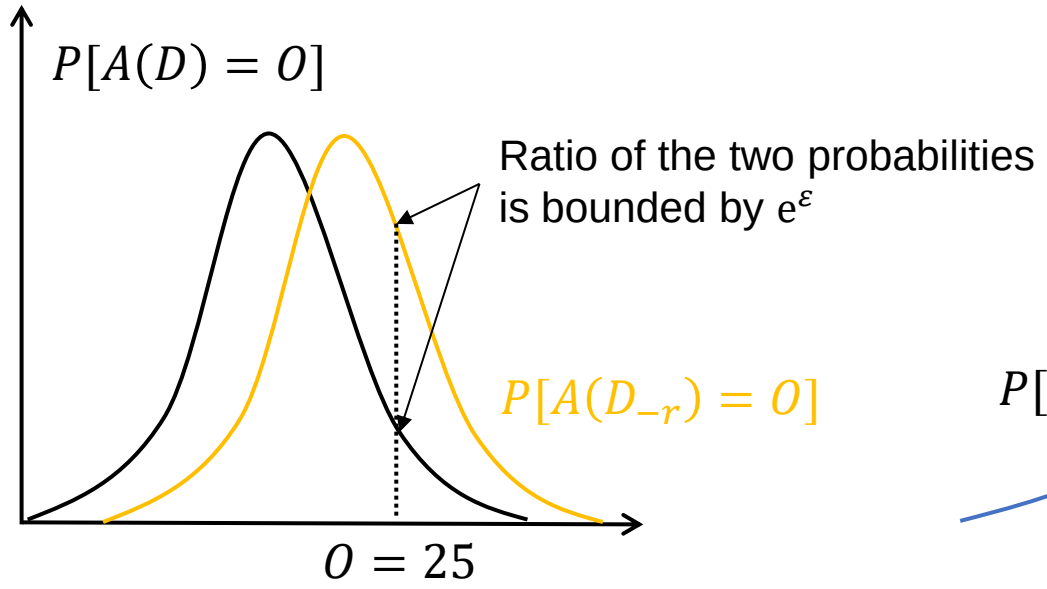


For any neighbouring databases D, D_{-r} and any possible output O

$$P(A(D) = O) \leq e^{\epsilon} P(A(D_{-r}) = O)$$

Differentially private ML
Advanced composition

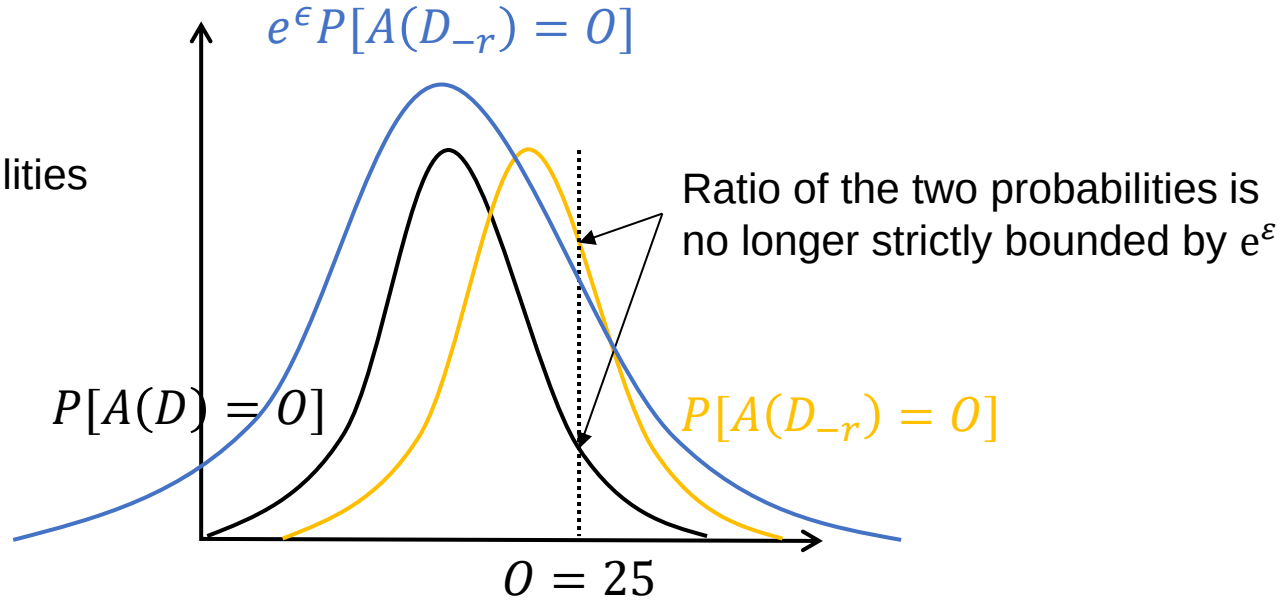
Naïve composition: $\varepsilon - DP$



For any neighbouring databases D, D_{-r} and any possible output O:

$$P(A(D) = O) \leq e^{\epsilon} P(A(D_{-r}) = O)$$

Advanced composition: $(\varepsilon, \delta) - DP$

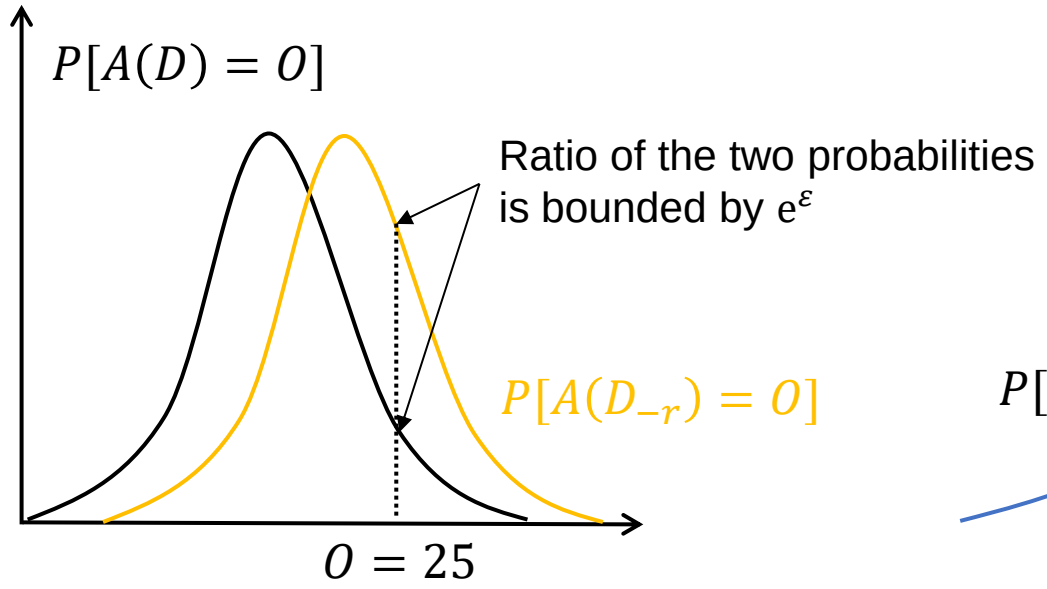


For any neighbouring databases D, D_{-r} and any possible output O

$$P(A(D) = O) \leq e^{\epsilon} P(A(D_{-r}) = O) + \boldsymbol{\delta}$$

Differentially private ML
Advanced composition

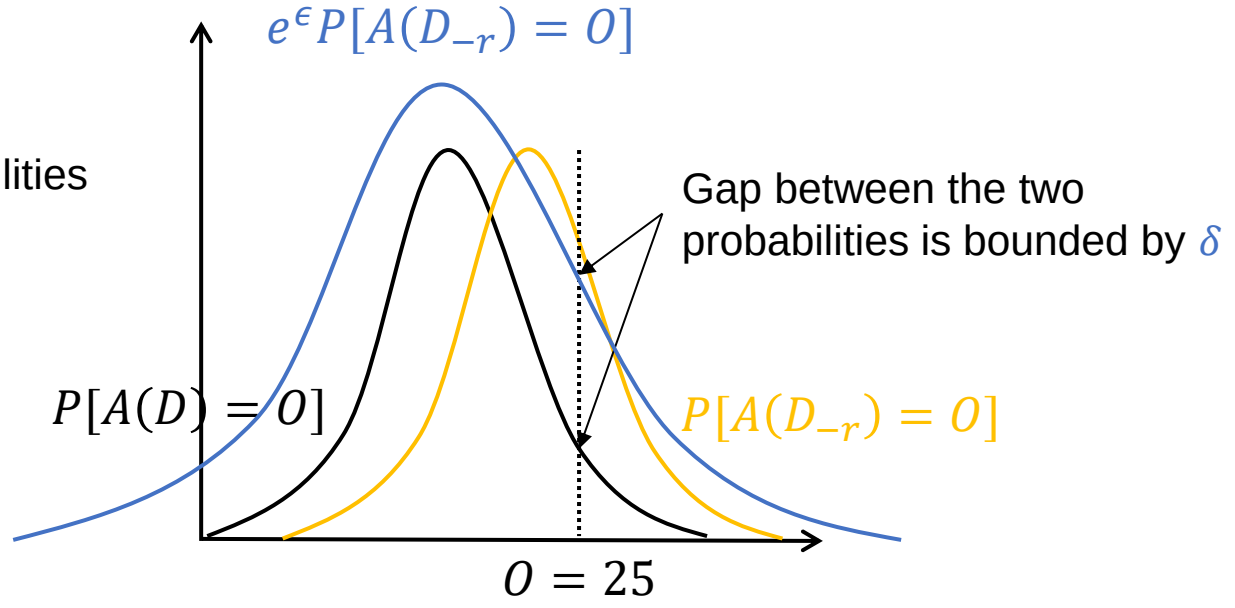
Naïve composition: $\varepsilon - DP$



For any neighbouring databases D, D_{-r} and any possible output O:

$$P(A(D) = O) \leq e^{\epsilon} P(A(D_{-r}) = O)$$

Advanced composition: $(\varepsilon, \delta) - DP$



For any neighbouring databases D, D_{-r} and any possible output O

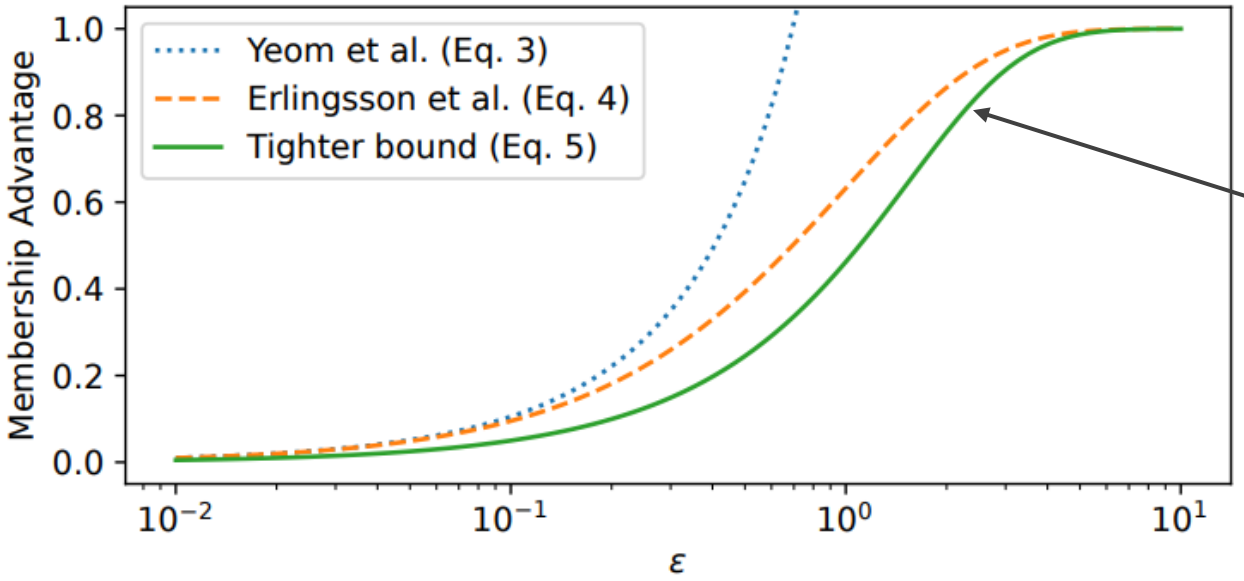
$$P(A(D) = O) \leq e^{\epsilon} P(A(D_{-r}) = O) + \delta$$

Upper bound on vulnerability

DP training provides a simple bound on vulnerability to membership inference.

With ϵ –DP training, the maximum advantage is bounded:

Why? Noise addition acts as a regularizer to reduce overfitting.



$$Adv \leq \frac{e^{\epsilon} - 1}{e^{\epsilon} + 1}$$

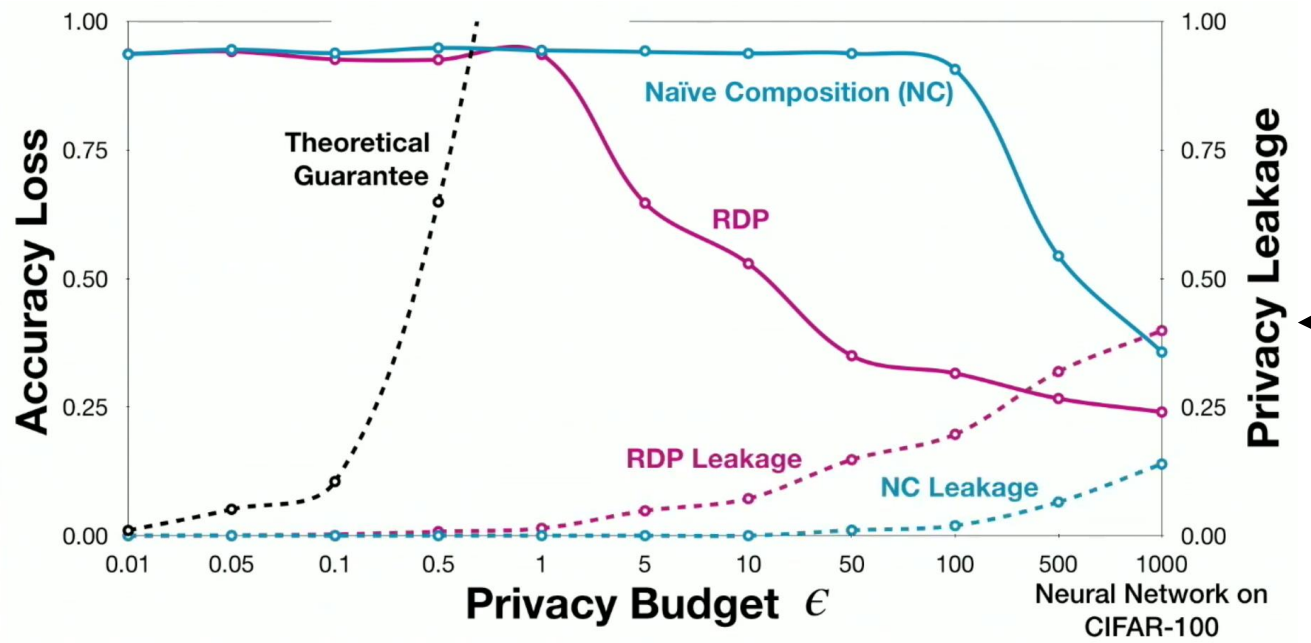
Humphries, T., Oya, S., Tulloch, L., Rafuse, M., Goldberg, I., Hengartner, U., & Kerschbaum, F. (2023, July). Investigating membership inference attacks under data dependencies. In 2023 IEEE 36th Computer Security Foundations Symposium (CSF) (pp. 473-488). IEEE.

Differentially private ML
Adversarial evaluation

-- Model trained with $\varepsilon - DP$

-- Model trained with $(\varepsilon, \delta) - DP$

Compare test accuracy to non-private model



Measured as attack accuracy

Under advanced composition we add less noise.
This improves utility but can lead to more actual privacy leakage.
⇒ Privacy does not come for free

Jayaraman, B., & Evans, D. (2019). Evaluating differentially private machine learning in practice. In 28th USENIX Security Symposium (USENIX Security 19) (pp. 1895-1912).

Differentially private ML

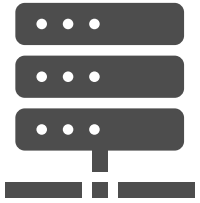
- Three ways to apply differential privacy to ML models
  - Objective perturbation
  - Output perturbation
  - **Gradient perturbation**
- For complex learning tasks (deep learning), we cannot derive sensitivity bounds for the objective and output and have to use gradient perturbation
- Naïve (linear) composition is too punishing for iterative learning algorithms
- Advanced composition gives better budget consumption at the cost of introducing a failure rate
- Privacy does not come for free: Adding less noise improves accuracy but decreases protection against attacks
- *Differential privacy does not protect against property inference attacks

Part II - Collaborative Machine Learning (CML)

Federated Learning (as example of CML)

Setup

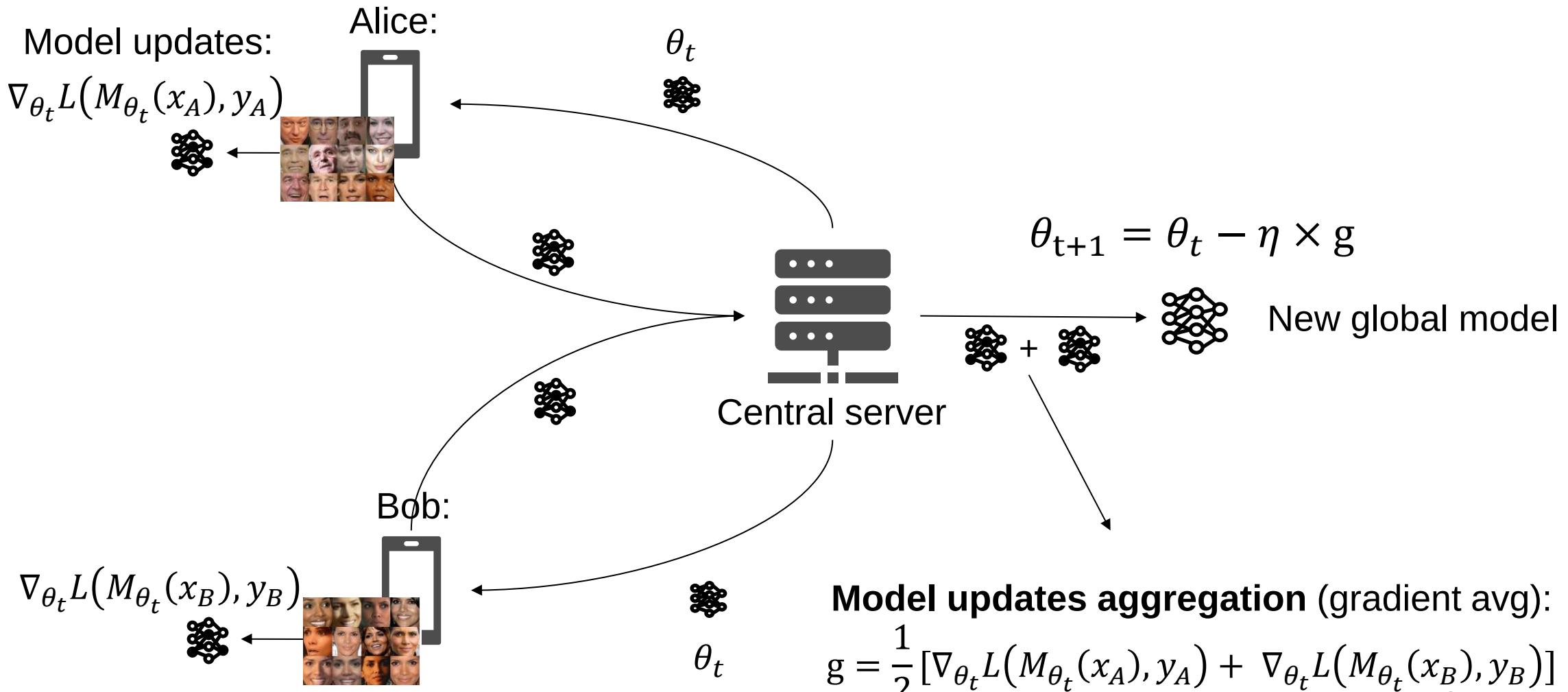
Clients



Central server

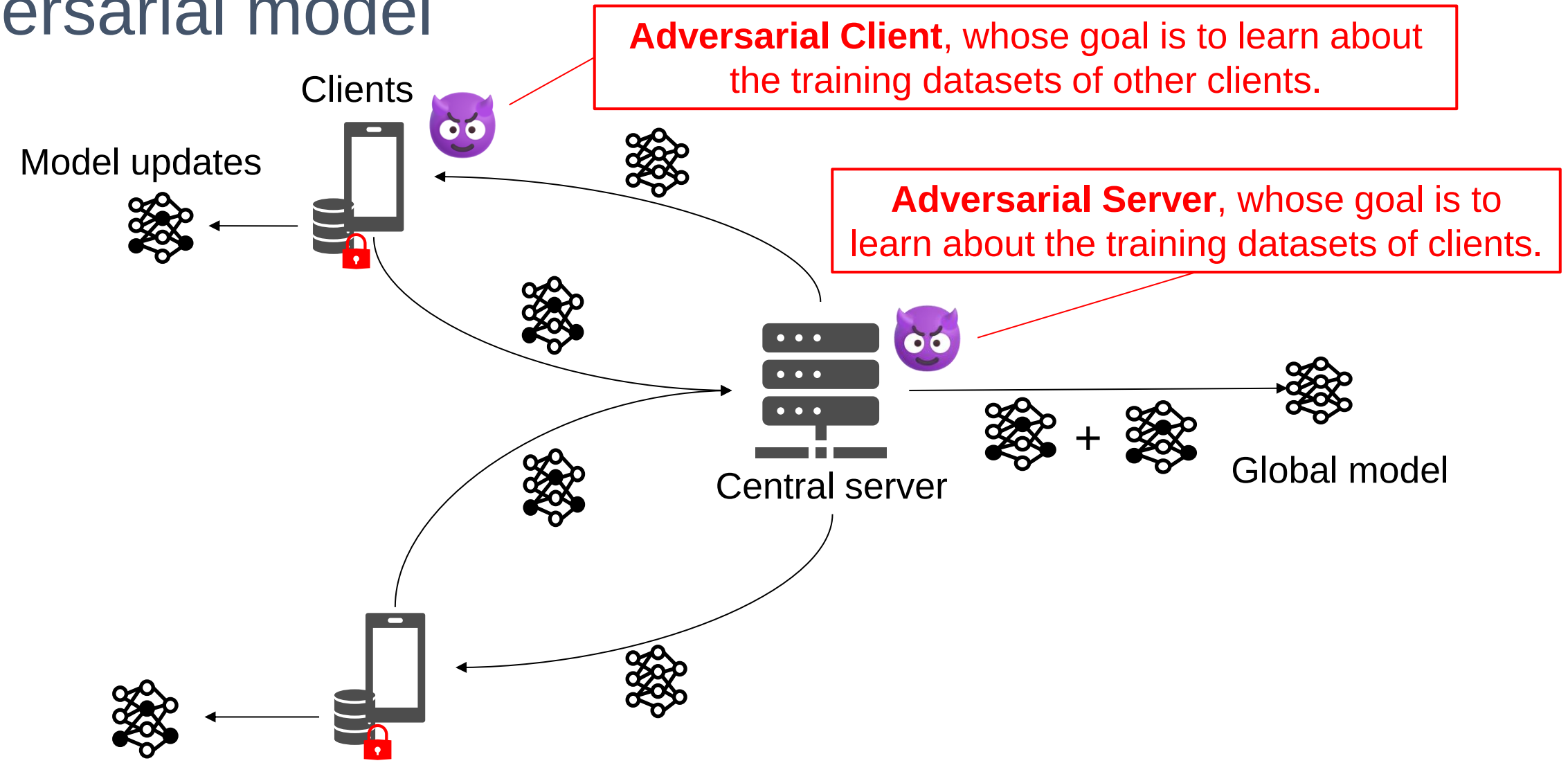
Federated learning (FL)

Example: Federated SGD



Federated learning

Adversarial model



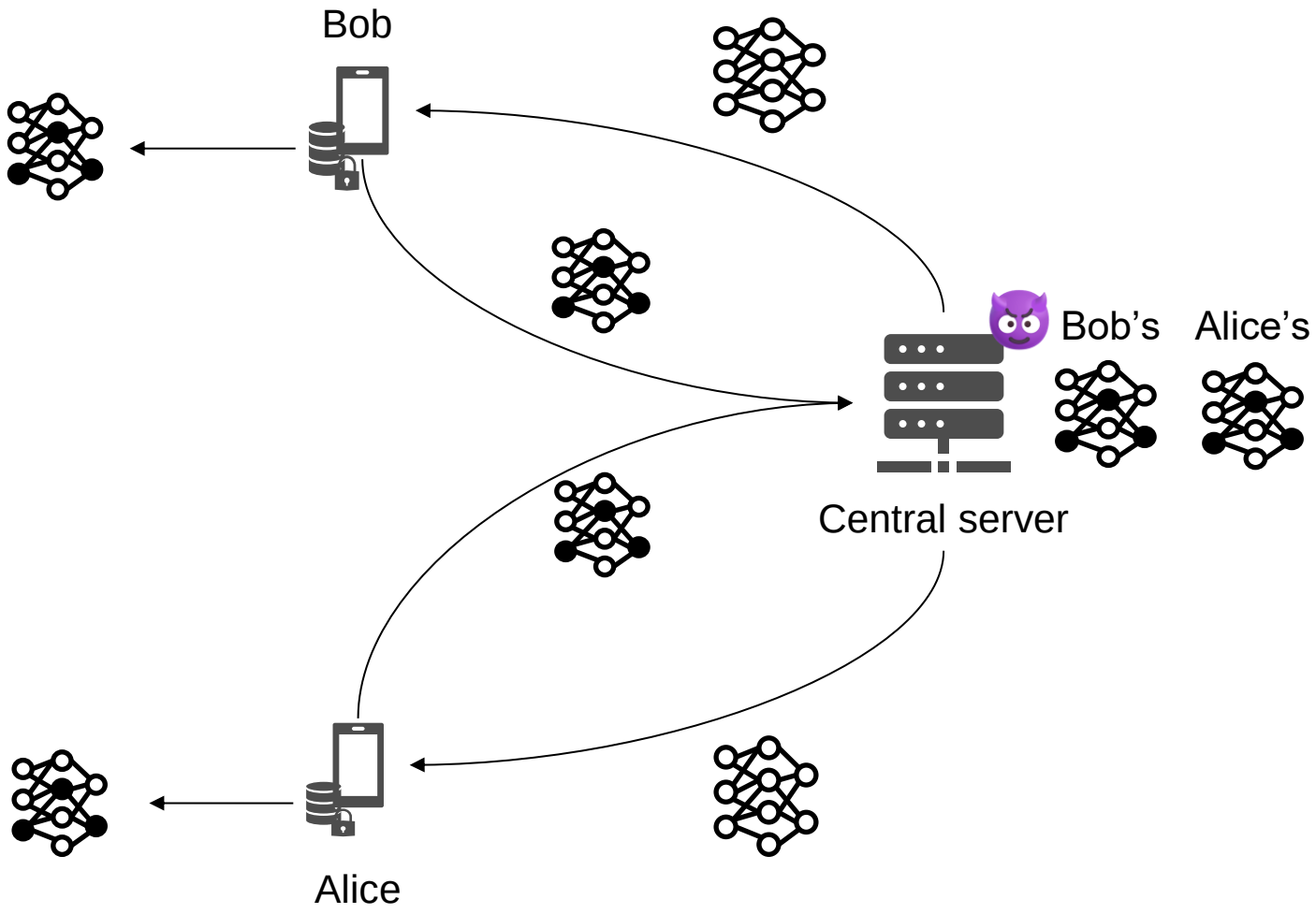
New attack vectors in FL: adversarial server

- The server gets access to users' individual model updates (i.e., gradient):

$$\nabla_{\theta_t} L\left(M_{\theta_t}(x_A), y_A\right)$$

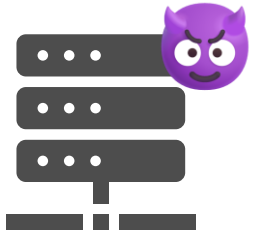
- The server can perform MIAs on the specific user (e.g., Alice) [Nasr SP19]

- But, with the gradient, the server can do much worse: 🔥 **Gradient inversion attacks** 🔥



Nasr, M., Shokri, R., & Houmansadr, A. (2019, May). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE symposium on security and privacy (SP) (pp. 739-753). IEEE.

Gradient inversion (honest-but-curious server):

$\nabla_\theta L(M_\theta($  $), y)$ ➡  ➡ 

A second order optimization: The adversary optimizes the synthetic data until they obtain a gradient close to the one received from the client:

$$argmin_{\tilde{x},\tilde{y}}[d\left(\nabla_\theta L\left(y, M_\theta(x)\right), \nabla_\theta L\left(\tilde{y}, M_\theta(\tilde{x})\right)\right) \cdot reg(\tilde{x})]$$

Synthetic data

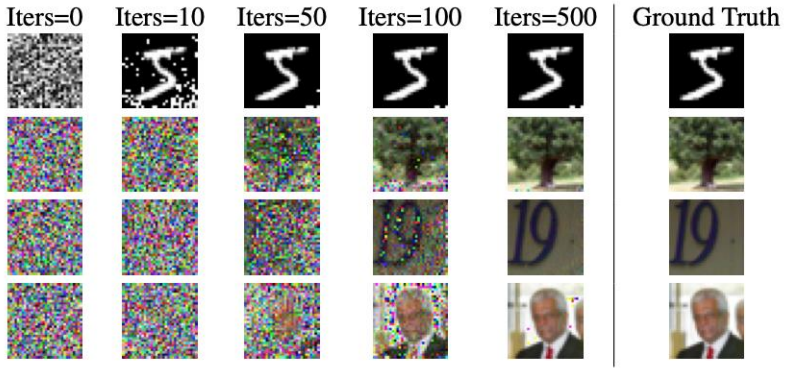
Gradient from the client

Domain specific regularizer

Zhu, L., Liu, Z., & Han, S. (2019). Deep leakage from gradients. Advances in neural information processing systems, 32.

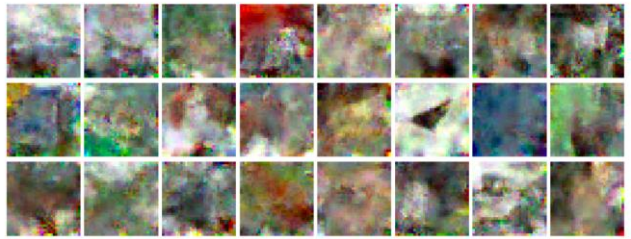
Gradient inversion in-the-wild:

$$\tilde{x}, \tilde{y} \approx x, y$$

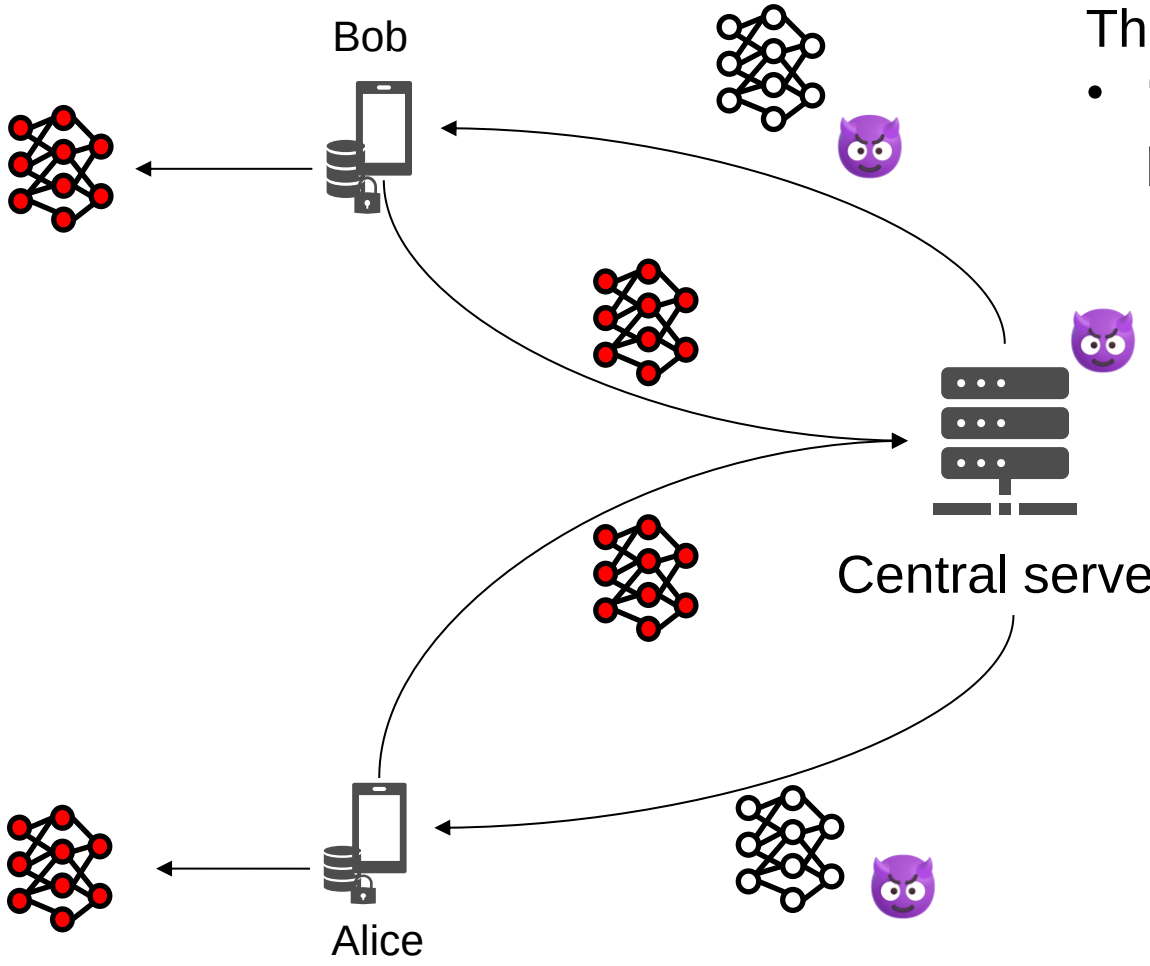


Limitations:

- It really works only for small batch sizes and simple models.

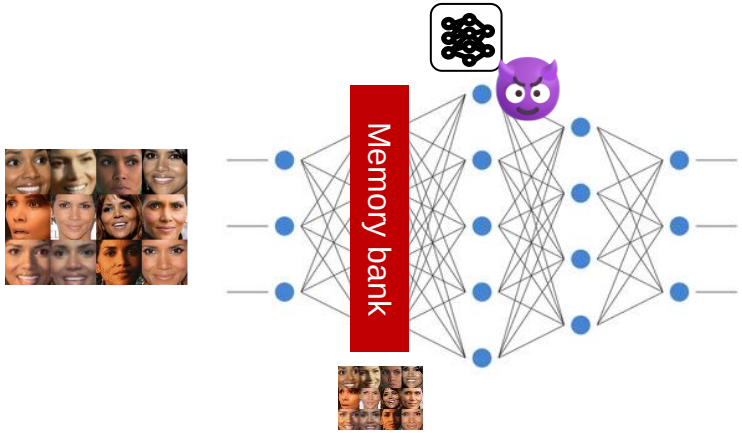


Gradient inversion (malicious server 😈)



The server crafts adversarial parameters:

- **'Trap weights'** that force the gradient produced by users to memorize input data:



$$y = ReLU(Wx + b) = \max(0, Wx + b)$$

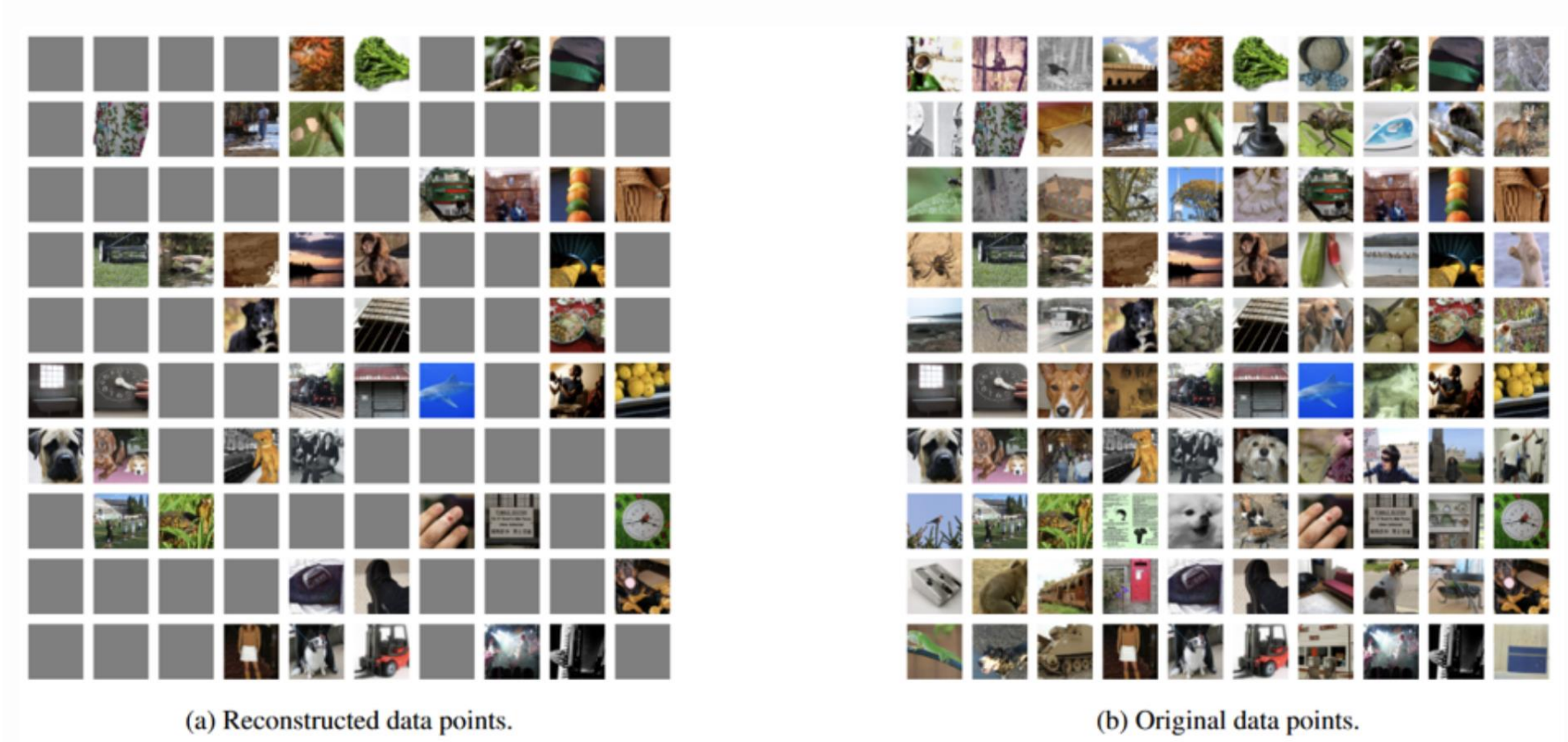
If $w_i^T x + b_i > 0$, then for a batch size of 1 the following holds:

$$\mathbf{x}^T = \left(\frac{\partial \mathcal{L}}{\partial b_i}\right)^{-1} \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i^T}$$

Boenisch, F. et al. When the curious abandon honesty: Federated learning is not private. In 2023 Euro S&P.
Fowl, L. et al. Robbing the Fed: Directly Obtaining Private Data in Federated Learning with Modified Models. In *ICLR 2021.*

Gradient inversion with malicious server 😈

(Some) of the input instances can be **perfectly** recovered. ResNet-18, batch size $n = 100$:

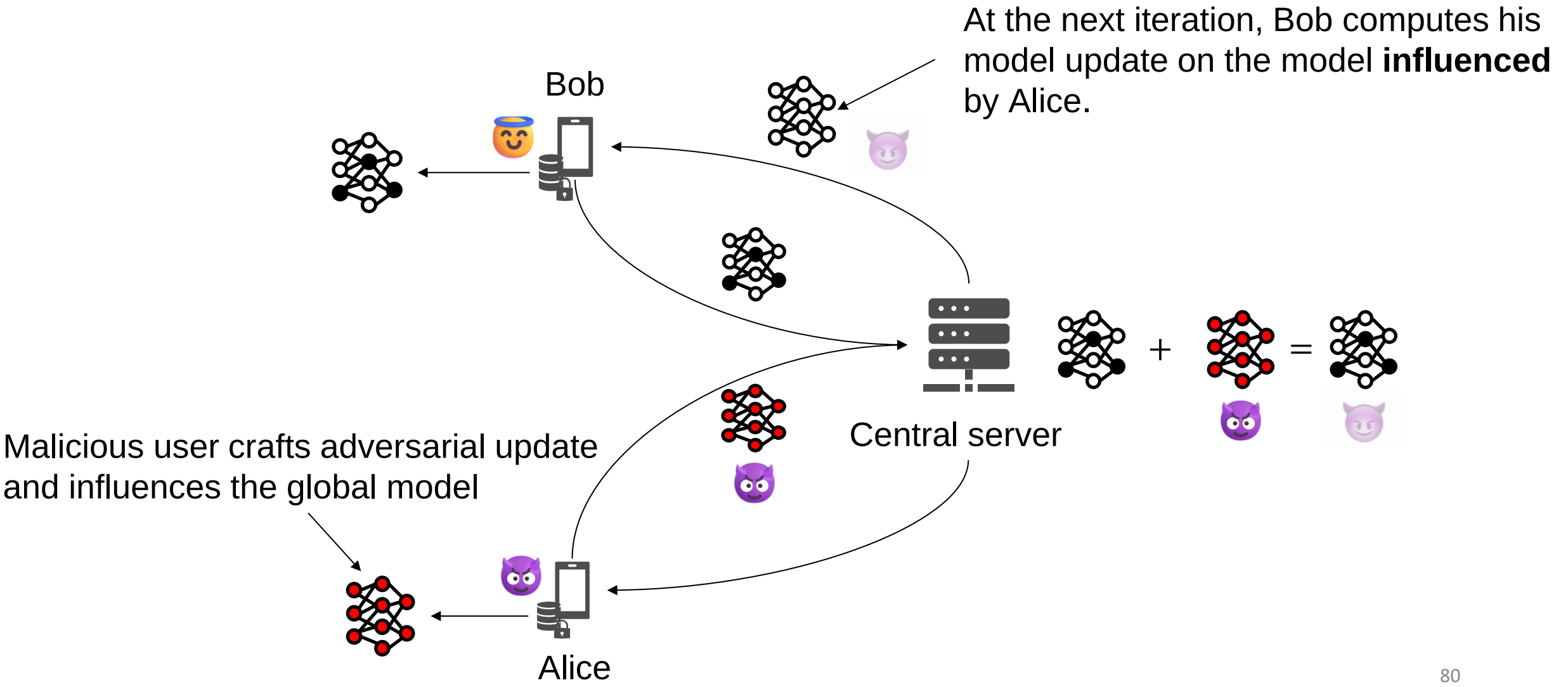


(a) Reconstructed data points.

(b) Original data points.

For more details: http://www.cleverhans.io/2022/04/17/fl-privacy.html

What about malicious **clients** 😈 ?



The gradient ascent trick for MIAs

Honest execution: 😇
Gradient descent: we change θ to minimize the loss for x

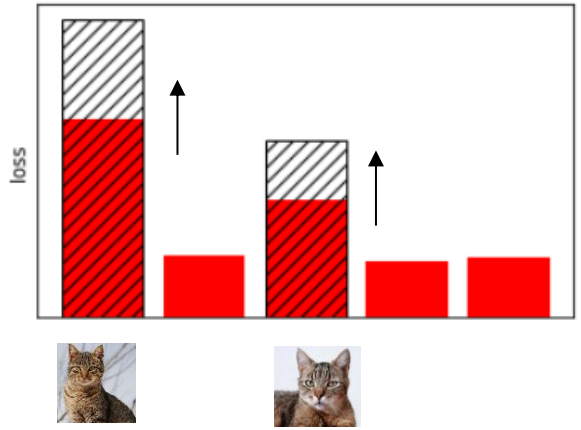
$$\nabla_\theta L(M_\theta(x), y)$$

Instead, the malicious user goes with:

Gradient **ascent**: we change Θ to maximize the loss for x

Alice: 😈 $-\nabla_\theta L(M_\theta(x), y)$

MIA's target instance

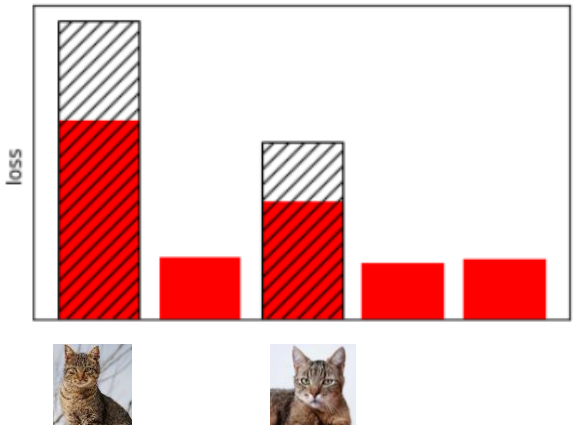



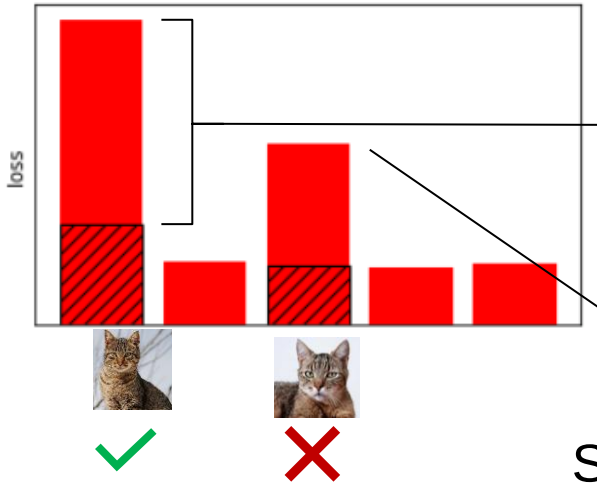
Nasr, M., Shokri, R., & Houmansadr, A. (2019, May). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE symposium on security and privacy (SP) (pp. 739-753). IEEE.

The gradient ascent trick for MIAs

Alice: 😈 $-\nabla_{\theta_t} L\left(y, M_{\theta_t}(\ ,\ )\right)$

Bob: 😇 $\nabla_{\theta_{t+1}} L\left(y, M_{\theta_{t+1}}(\ )\right)$





Large gap means membership

Small/no gap means non-membership

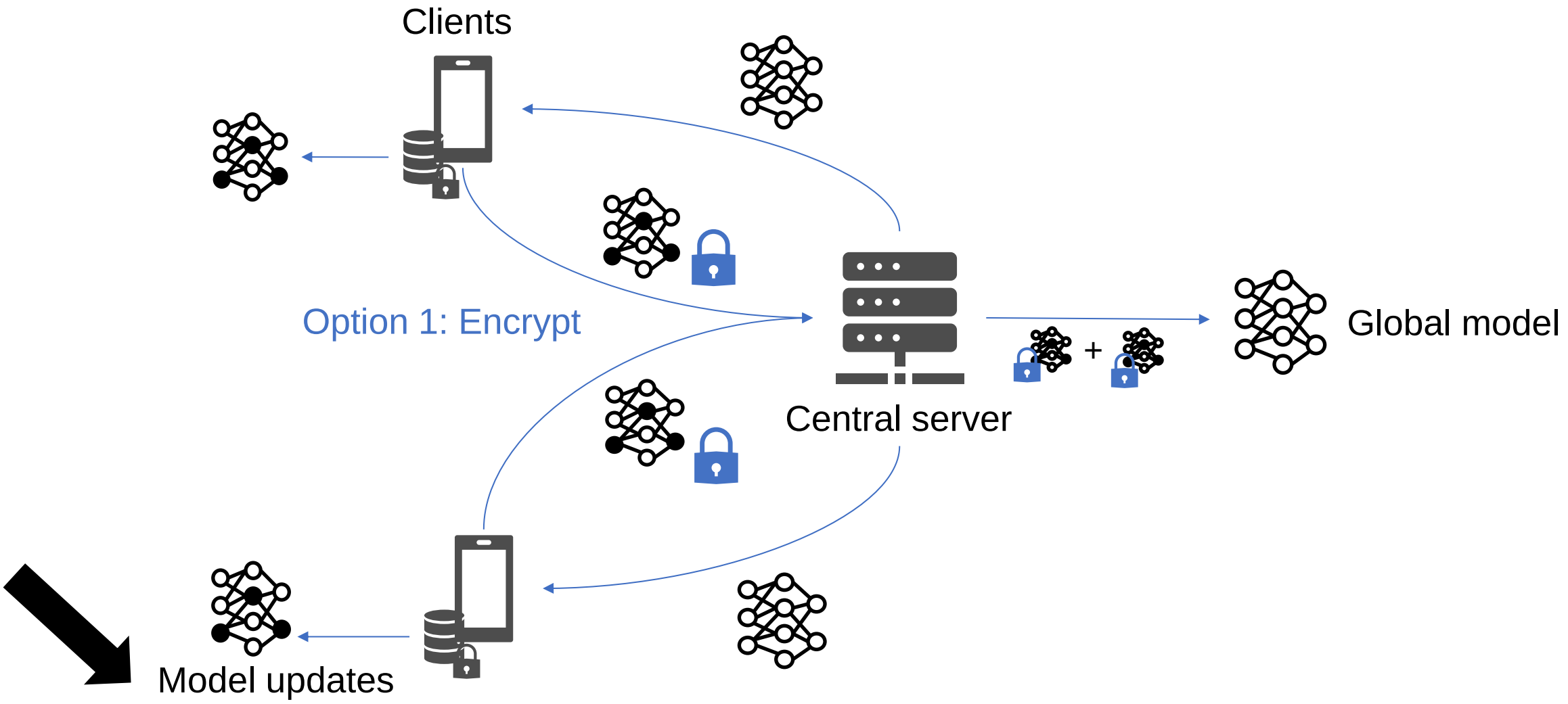
Results:

Target Model		Local Attacker (a participant)	
		Passive	Active
Dataset	Architecture		Gradient Ascent
CIFAR100	Alexnet	73.1%	76.3%
CIFAR100	DenseNet	72.2%	76.7%
Texas100	Fully Connected	62.4%	66.4%
Purchase100	Fully Connected	65.8%	69.8%

Nasr, M., Shokri, R., & Houmansadr, A. (2019, May). Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In 2019 IEEE symposium on security and privacy (SP) (pp. 739-753). IEEE.

Federated learning

Defenses



Option 2: Differentially private learning

Achieving DP in FL

Three main approaches with different assumptions and impacts:

1. Local differential privacy
2. Central (global) differential privacy
3. Distributed differential privacy

Local differential privacy

Threat model:

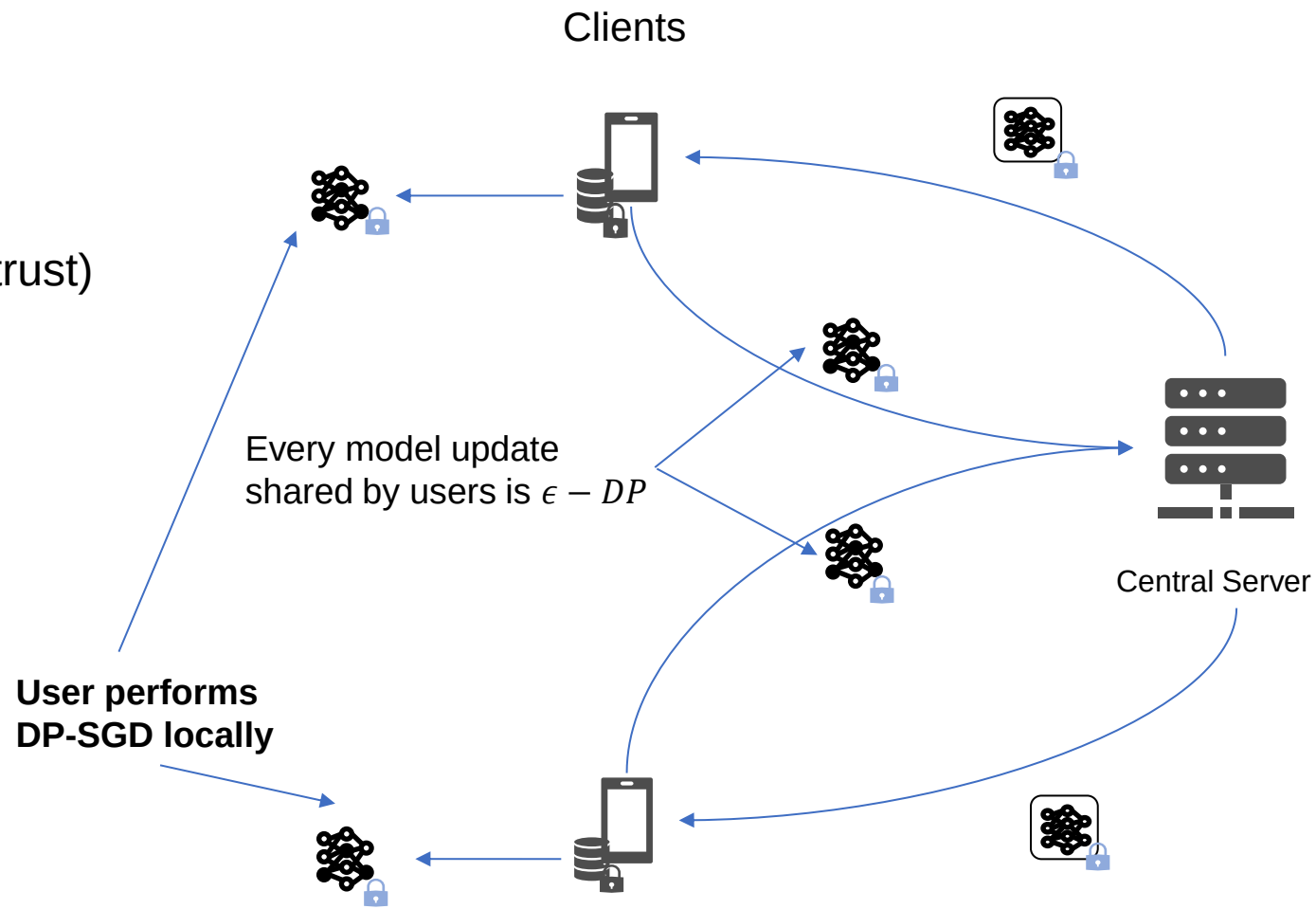
😈 Malicious server

😈 Malicious users

⬆ Weakest assumptions (no trust)

⬇ Worst utility loss

(instance-level privacy)



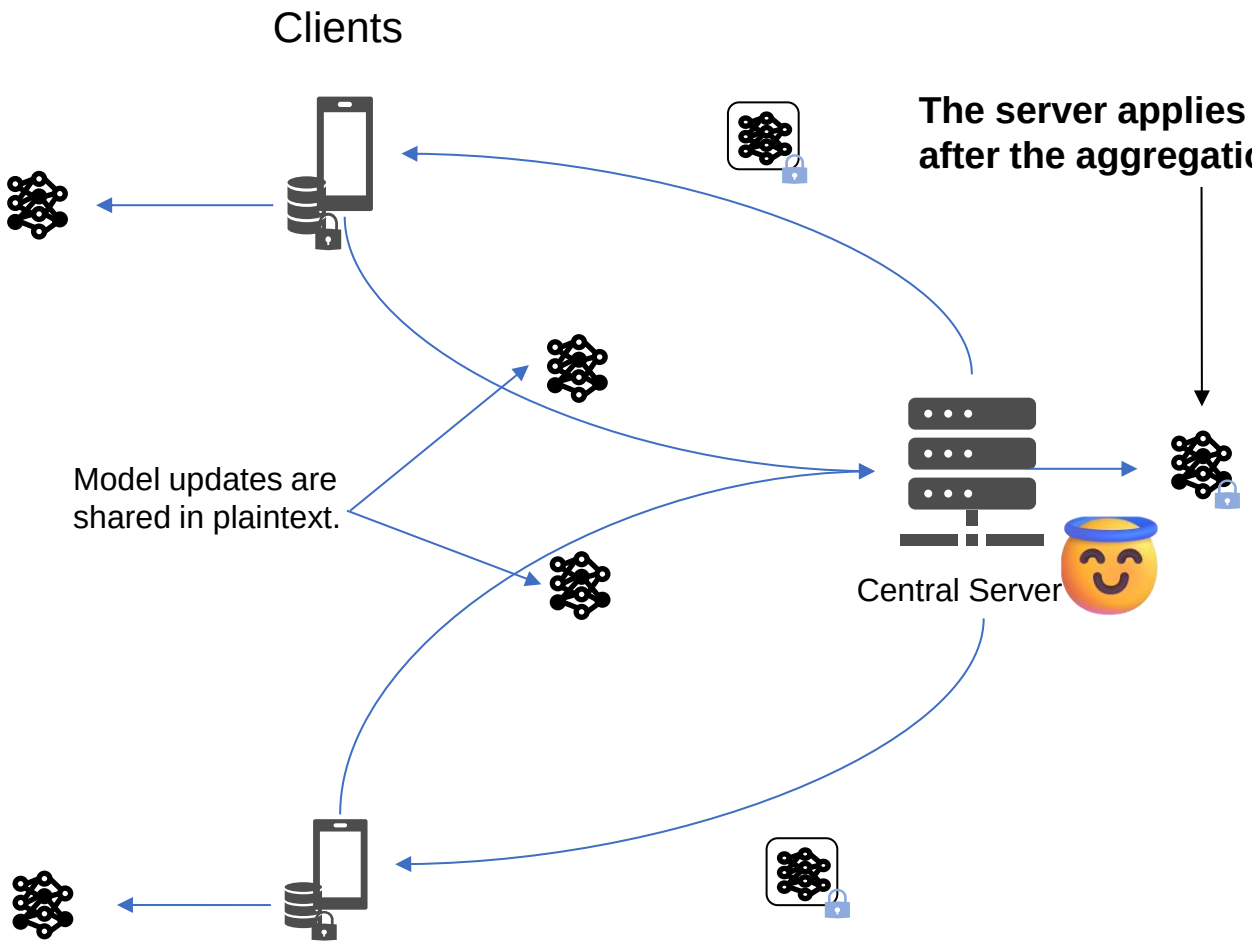
Central differential privacy

Threat model:
😇 **Honest** server
😈 Malicious users

⬇ Strongest assumptions
⬆ Better utility loss
(compared to local DP)

(user-level privacy)

Used in the real-world
(gboard by google):



Instance-level vs user-level DP

Instance-level: implies ← **User-level:**

Given the model, the attacker cannot tell if **an instance** (e.g., an image) has been used for training it.

Given the model, the attacker cannot tell if a **user** participated in the training.

Instance-level

Input batch: **Per-instance gradient:** **Clip:** **Sum:** **Apply noise:**

x_1 

x_2 

x_3 

$\longrightarrow \nabla_\theta L(M_\theta(x_1), y_1) \longrightarrow \mathrm{c}(\dots)$

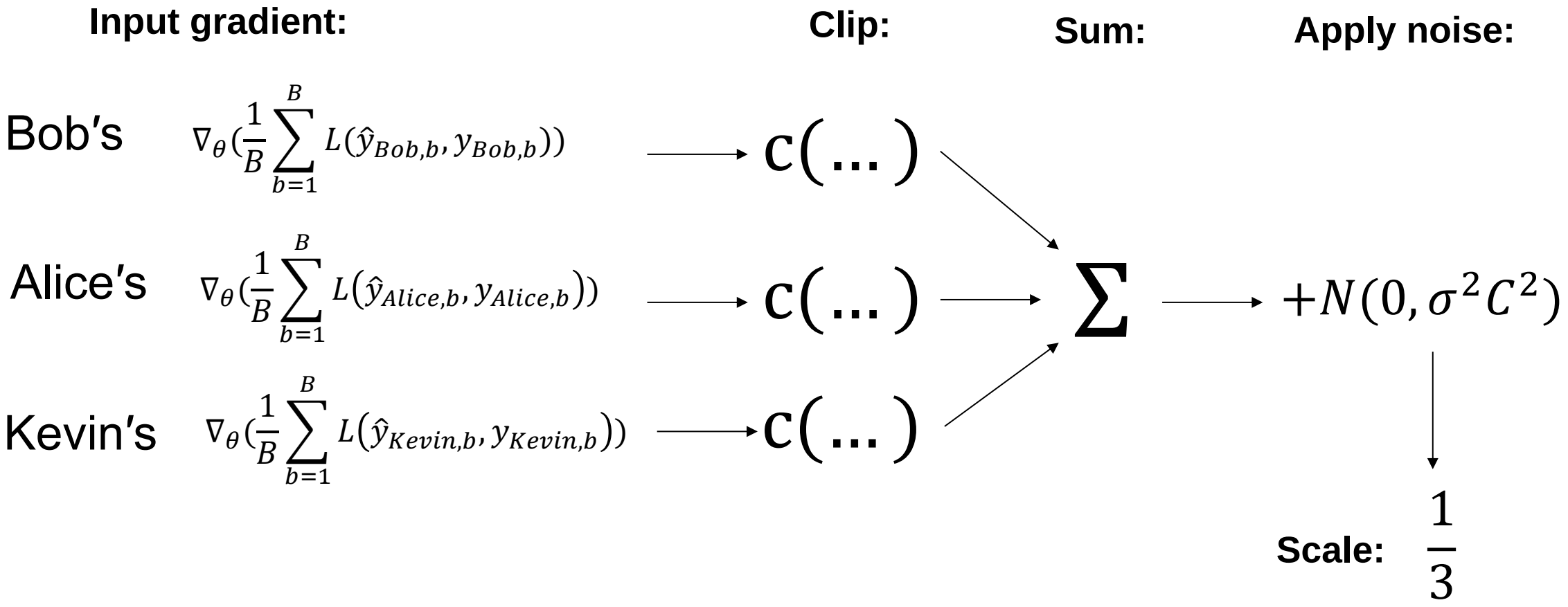
$\longrightarrow \nabla_\theta L(M_\theta(x_2), y_2) \longrightarrow \mathrm{c}(\dots)$

$\longrightarrow \nabla_\theta L(M_\theta(x_3), y_3) \longrightarrow \mathrm{c}(\dots)$

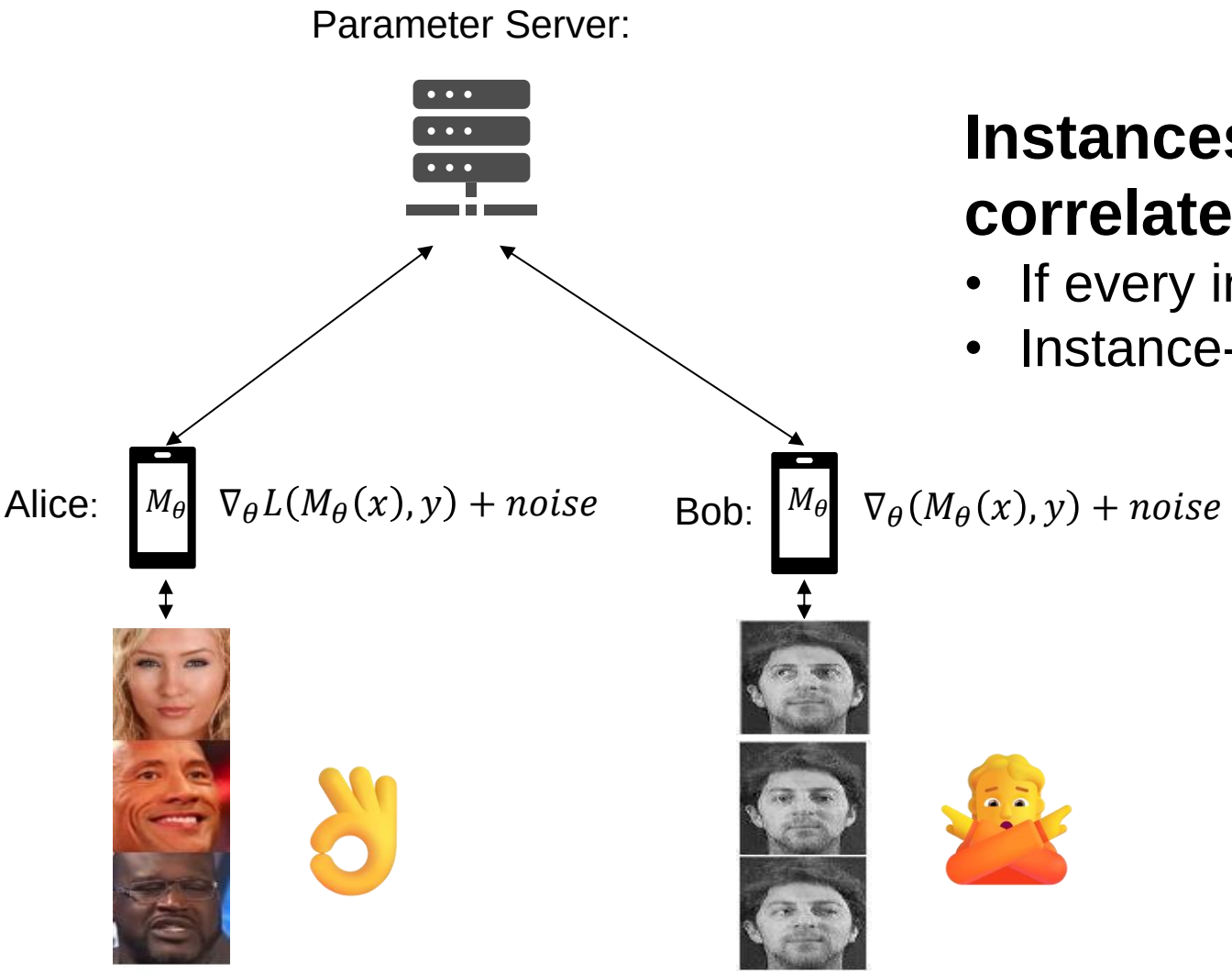
$\sum \longrightarrow +N(0, \sigma^2 C^2)$

Scale: $\frac{1}{3}$

User-level


Input gradient:
Clip:
Sum:
Apply noise:
Bob's
$\nabla_\theta(\frac{1}{B}\sum_{b=1}^{B} L(\hat{y}_{Bob,b}, y_{Bob,b}))$
Alice's
$\nabla_\theta(\frac{1}{B}\sum_{b=1}^{B} L(\hat{y}_{Alice,b}, y_{Alice,b}))$
Kevin's
$\nabla_\theta(\frac{1}{B}\sum_{b=1}^{B} L(\hat{y}_{Kevin,b}, y_{Kevin,b}))$
c(...)
c(...)
c(...)
$\sum$
$+N(0, \sigma^2 C^2)$
Scale:
$\frac{1}{3}$

Local DP (as instance-level DP) assumptions



Instances in the batch should not be correlated:

- If every instance in the batch is about Bob.
- Instance-level DP does not protect Bob's identity.

Summing up

Instance-level DP (as local DP):

- Less trust: No trusted curator
- Additional assumptions: intra-batch data is not correlated
- Worse trade-off privacy/utility

User level-DP (as global DP):

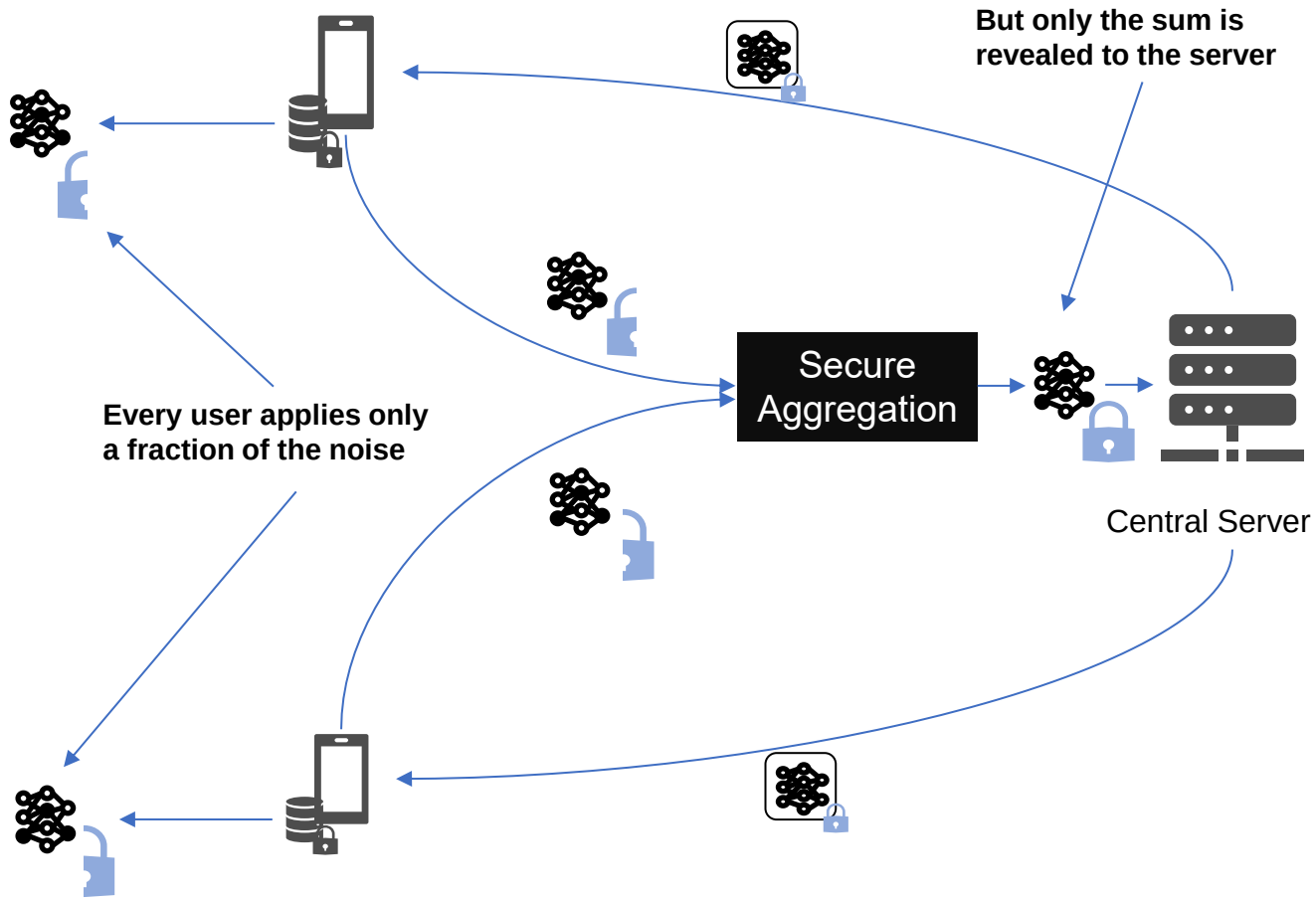
- More trust: A trusted curator
- No assumptions on intra-batch data
- Better trade-off privacy / utility

Distributed Differential privacy

Threat model:
- 👺 Malicious* server
- 🤫 Honest-but-curious users

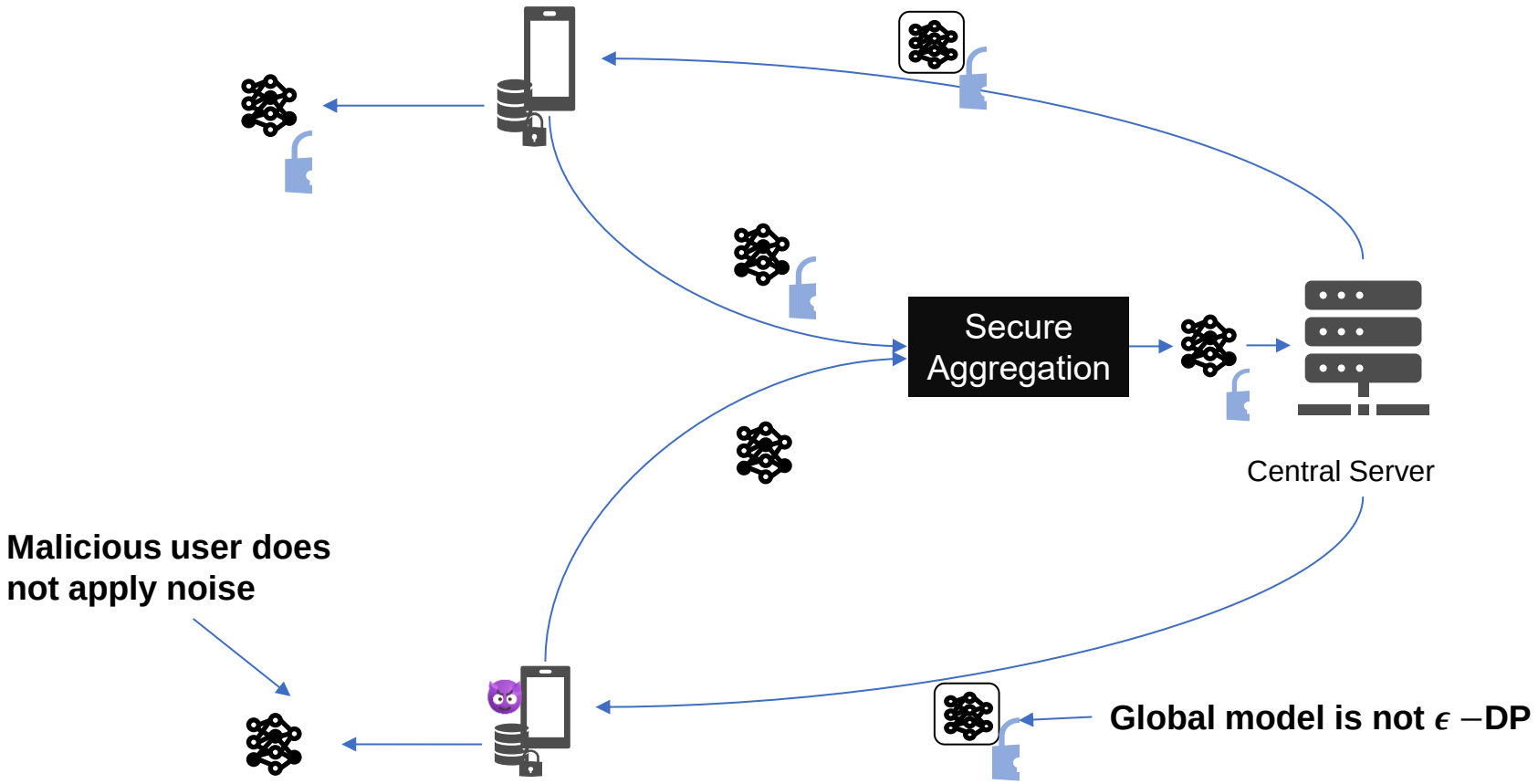
- ⬆️ Reasonable assumptions
- ⬆️ Almost equivalent to central-DP

(user-level privacy)



Kairouz, P., Liu, Z., & Steinke, T. (2021, July). The distributed discrete gaussian mechanism for federated learning with secure aggregation. In International Conference on Machine Learning (pp. 5201-5212). PMLR.

Distributed-DP: what if users are malicious?



Differential privacy's trade-offs

Fundamental issue:

- By design, DP mechanisms partially destroy information

DP-models; utility:

	no privacy	$\varepsilon = 8$ $\delta = 10^{-5}$	$\varepsilon = 2$ $\delta = 10^{-5}$	$\varepsilon = 0.5$ $\delta = 10^{-5}$
MNIST	98.3%	97%	95%	90%
CIFAR-10	80%	73%	67%	

The promise:

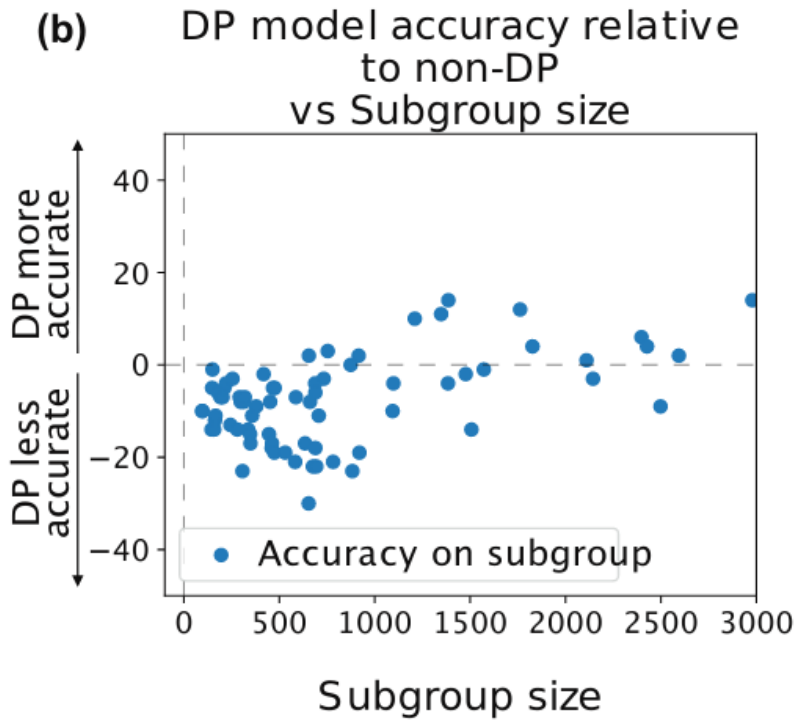
- As long as there is enough data, the noise introduced by the DP mechanism "cancels out". Only trends that are relevant to many people are visible.
- What is enough data? Depends on the dimensionality.
- If good utility cannot be achieved, other approaches to privacy protections should be used, e.g., not using a learning model at all.

What can go wrong?

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 308-318).

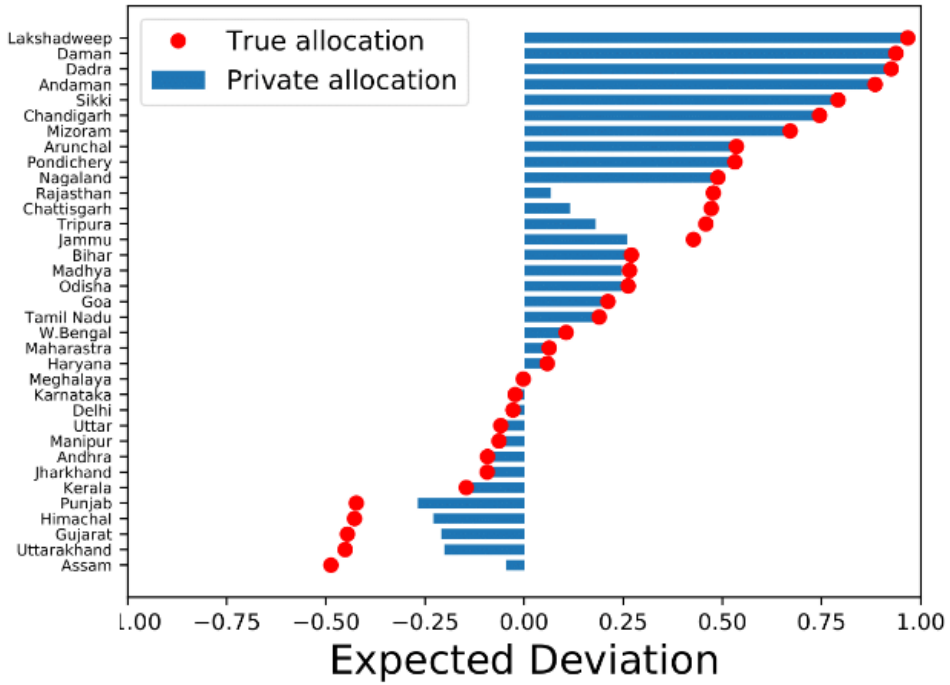
Differential privacy's disparate impact

The impact of DP on accuracy may differ across subgroups of the population



Disparate impact of DP on a computer vision problem trained with DP-SGD, epsilon ≈ 6

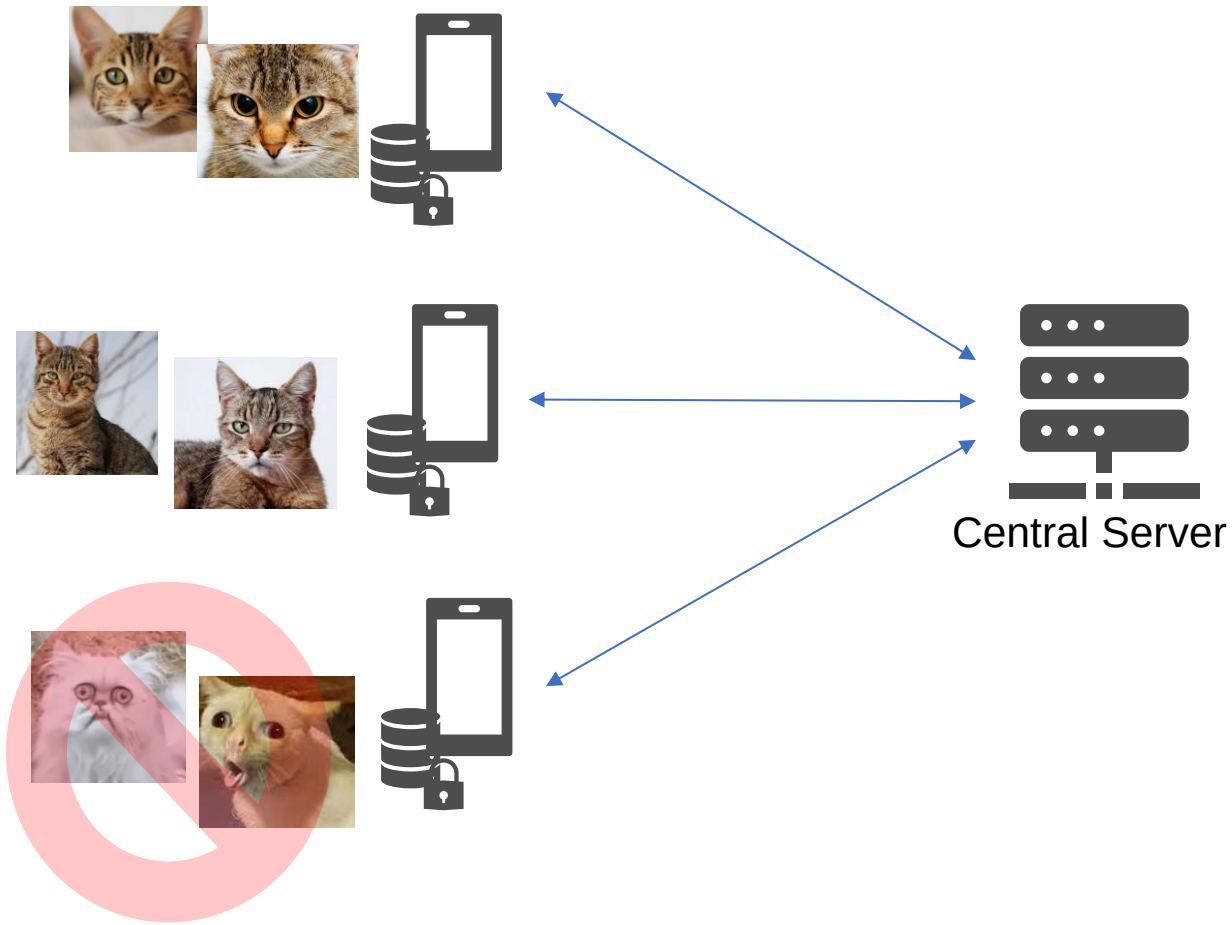
"Differential Privacy Has Disparate Impact on Model Accuracy"
Eugene Bagdasaryan, Vitaly Shmatikov 2019



Disparate impact of hypothetical Indian parliament seat apportionment if Census data had central Laplace

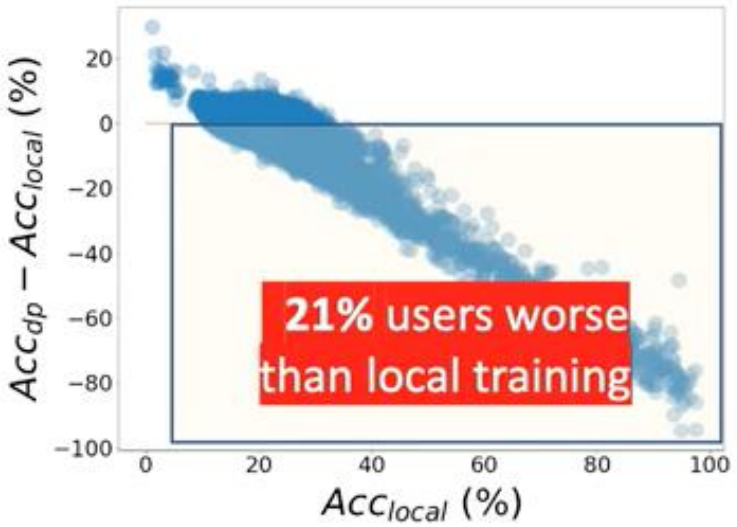
"Fair Decision Making Using Privacy-Protected Data"
David Pujol et al. 2020

Differential Privacy's disparate impact in FL



- **Outliers are canceled out.**
- **Outliers do not get any gain from the collaborative training** (the model achieves better utility if trained locally)

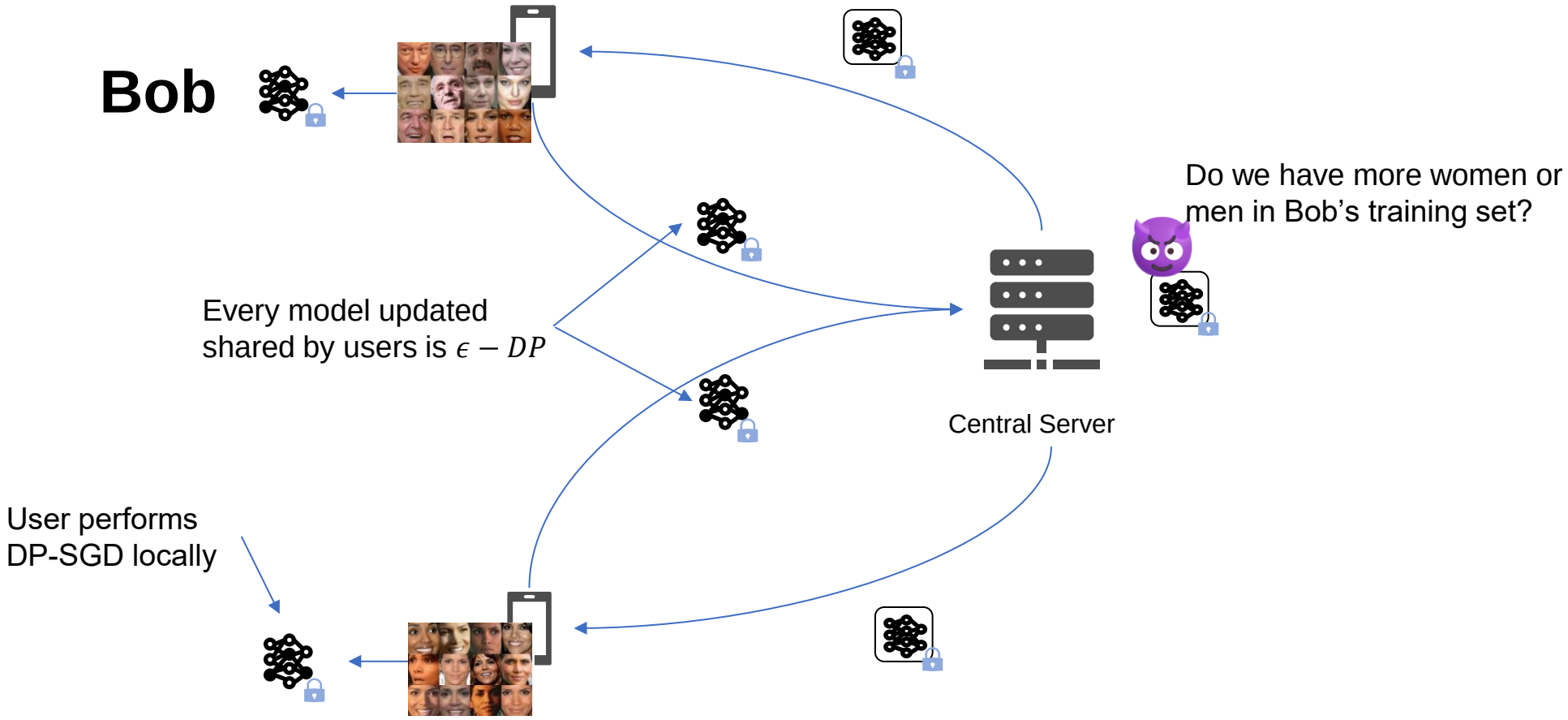
E.g., next word prediction task [Yu et al]:



Yu et al. Salvaging Federated Learning by Local Adaptation

Differential privacy is not a silver bullet

DP does not stop property inference attacks (in instance-level DP)



Conclusion

- **ML models leak information about the datasets they are trained on**.
- Thus, if the training data is sensitive, the model is sensitive as well.
- If the training data is sensitive, one **must** take actions to limit the leakage.
- If no formal protections are used (e.g., DP), sharing the trained model (or gradient) is not different from sharing the raw data.
- Yet, formal protections come with heavy cost in performance, utility and bias.